
Robonauts • LEGO Bots • Combat Bots
SERVO
FOR THE ROBOT INNOVATOR
www.servomagazine.com
MAGAZINE
May 2011
MEET THE NEW
LUNAR ROVER
From MINDS-i
✦ Build A Single Axis Robot Arm
This Space Shuttle inspired version uses a snare to pick things up.
✦ A UART/SPI Monitor For Your Micro
Find out what's going on inside your controller with this super snooper.
U.S. $5.50 CANADA $7.00
05
0 71486 02422 4

INVEST in the BEST

YOUR ROBOT WILL THANK YOU

At Hitec, we know how much time and money you dedicate to your robot hobby. So why not make sure your servos are delivering what they promise? Our technologically advanced mega servos are tough enough for even your most challenging projects. Designed with our newest high resolution "G2.5" 12-bit programmable digital circuit and indestructible titanium gears, the HS-7980TH and HS-M7990TH give mega torque and speed with pinpoint accuracy. Built to last, these servos bring unprecedented power and sustainability to your investment.

Our HS-7980TH employs durable mechanical potentiometer technology while the HS-M7990TH utilizes the first ever magnetic encoder.

	6 Volts		7.4 Volts				
Model	**Speed**	**Torque**	**Speed**	**Torque**	**Part#**	**Dimensions**	**Weight**
HS-7980TH	0.20	500 oz.in	0.17	611 oz.in	37980S	1.72 x 0.88 x 1.57 in	2.70 oz
HS-M7990TH	0.20	500 oz.in	0.17	611 oz.in	37990S	1.72 x 0.88 x 1.57 in	2.70 oz


do more wire less
Wixel
USB. Wireless. Programmable. All for $19.95.
Introducing the Wixel, a programmable microcontroller module with integrated USB and a 2.4 GHz radio. Write your own program or load pre-compiled, open-source apps to give your next project a wireless serial link, create a remote sensor network, and so much more.
actual size
Pololu
Engage Your Brain
learn more at www.pololu.com/wixel

SERVO MAGAZINE

05.2011

VOL. 9 NO. 5

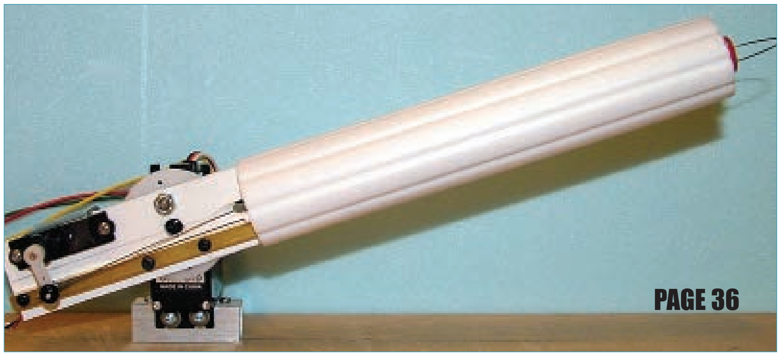
Did you know that each article in *SERVO Magazine* has its own webpage? It's where you go for downloads, comments, updates, corrections, or to link to the article in the digital issue. The unique link for each webpage is included with the article.

Columns

PAGE 10

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...


PAGE 36

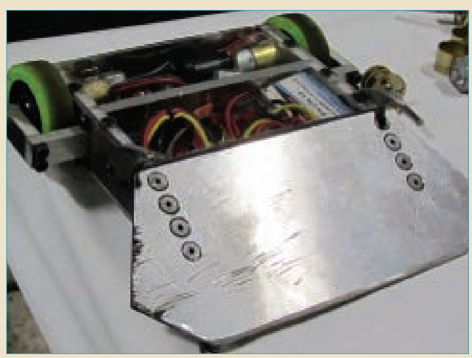

PAGE 22

Departments

The Combat Zone...

Features

Events



Mind / Iron

by Bryan Bergeron, Editor

Anthropomorphic Computing

While the prowess of the IBM Watson super computer is old news, its appearance on the Jeopardy show is worth some reflection. The producers of the show — dealing with the need to pull in a home video audience — had to provide something other than a box for the Watson persona. In addition to the blue globe with strands of light, there was a robotic finger to press the button. Although a handicap for a computer that could simply connect via a serial cable to the button circuitry, the mechanical linkage was deemed necessary for the viewing audience to relate to the robot. That is, the consensus was that the finger was needed to make the computer seem more like a human competitor.

This anthropomorphizing of computing isn't new, but Watson served to highlight the relevance of the technique as necessary for the acceptance of computers and computer technology. Additional examples of this practice can be seen in applications ranging from telemedicine — the practice of medicine over the Internet — to teletutoring — instruction via the Internet.

For example, in telemedicine, a one-on-one conversation between a patient and physician requires little more than a Skype setup with an inexpensive webcam and broadband connection. For higher resolution connections, there are a variety of commercial teleconferencing services from the likes of Cisco and others.

Nonetheless, if you make the robotic trade shows, you're bound to come into contact with one of several robotic doctors on the market. These are little more than teleconferencing services on a human-sized robotic platform (Google 'Mobile Unit Robot Doctor'). There are no hands or arms, but the head is a flat panel, normally containing a video of the physician's head atop a torso. According to the marketing material from these developers, patients relate better to a roving doctor that has some semblance of a human versus a computer monitor by the bedside.

As another example, consider teletutoring. Need a private tutor to help you or someone in your family with math homework? No problem. There are teachers in India with a Ph.D. in mathematics ready to help via teleconferencing. No time to drive to a school to receive private music lessons? No problem there either. It's easy to find a guitar teacher online who's happy to provide private lessons via a Skype connection.

I don't have any problems interacting with my guitar instructor through a laptop, perhaps because I've developed a connection with him through personal interaction. However, some educators feel that the teaching process is more effective if the teacher has more of a physical presence. This is where robotics comes in.

There's a trial underway in 21 elementary schools in South Korea where English instruction is provided through remotely

FOR THE ROBOT INNOVATOR
SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert, Tom Carroll, David Geer, Dennis Clark, R. Steven Rainwater, Daniel Albert, Kevin Berry, Gregory Intermaggio, David Ward, Gordon McComb, Mark Mitchell, Paul Verhage, Mike Jeffries, Pete Smith, Bryce Woolley, Evan Woolley

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

controlled robots. Relatively inexpensive English teachers in the Philippines control the wheeled, three foot tall robots. Children see the avatar representation of the teacher's face on the flat panel head, and the teacher can see and hear students. Hopes are that the robots developed by the Korea Institute of Technology (KIST) prove less expensive and less problematic than English teachers. In many rural areas of Korea, it's difficult to entice English teachers to stick around, in part because pay is better with private students in the large cities.

According to preliminary reports, the children are more accepting of a robotic teacher than of a much less expensive webcam and monitor perched atop a desk. It remains to be seen whether the results justify the added costs of the anthropomorphic platform.

Do you have an idea for how robotics can enhance the human-computer interface? Perhaps an online recipe book that not only provides a list of ingredients and cooking instructions, but that can lend a hand with chopping the onions or moving heavy pots and pans? I'd consider one that loads and unloads the dishwasher. In all seriousness, robotics undoubtedly has a place in making computing more accessible and acceptable. The challenge is identifying the application areas most in need of an anthropomorphic interface. **SV**

BIO-FEEDBACK

Dear SERVO:

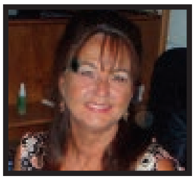
In the January '11 issue, Pete Smith wrote an article about a 2.4 GHz RC unit. Question: The receiver has three pins at the "Bat" position. What are their polarities? Many thanks.

S. Browman
Montreal

Response:

Sam, as per the photo below, the signal cable is nearest the label, and the ground furthest away with the red 5V in the center. Hope this helps.

Pete Smith

Robytes

by Jeff and Jenn Eckert

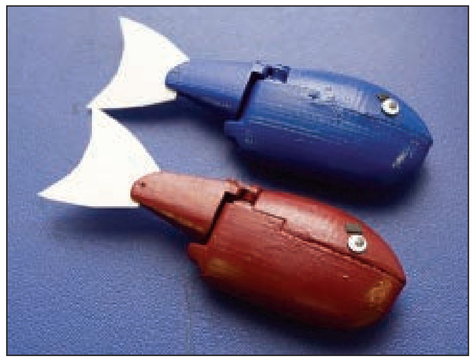
Hummingbot Demonstrated

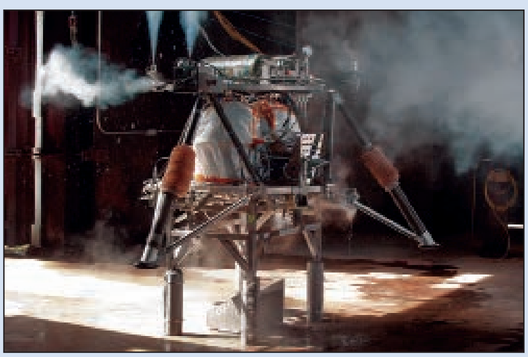

AeroVironment's Nano Hummingbird vehicle.

One of the many interesting projects at the Defense Advanced Research Projects Agency (DARPA, **www.darpa.mil**) is the Nano Air Vehicle program which involves the development of very small vehicles for both indoor and outdoor military missions. In Phase II of the project, AeroVironment, Inc. (**www.avinc.com**), has developed a hummingbird-like vehicle that is said to be the first ever to achieve "controlled precision hovering and fast-forward flight of a two-wing, flapping-wing aircraft that carries its own energy source and uses only the flapping wings for propulsion and control." The Nano Hummingbird can climb and descend vertically, fly sideways as well as forward and backward, and rotate both counter-clockwise and clockwise. The little bird has a wingspan of 16 cm and weighs only 19 g, including batteries, motors, communications systems, and a video camera. Under remote control, it achieved a series of required milestones including a hover endurance of better than eight minutes, 11 mph forward flight, and the ability to tolerate 5 mph wind gusts from the side. The NAV's intended duties haven't been spelled out in detail, but reconnaissance and surveillance are obvious applications.

Schooling the School

They may look like just some cheap fishing lures, but some fishbots developed at the Polytechnic Institute of New York University (**www.poly.edu**) actually represent a second generation of robots intended to help protect and preserve marine life. The units were developed by Maurizio Porfiri, assistant professor of mechanical engineering, in his quest to understand the schooling behavior of fish and to determine if robotic systems can be used to lure schools away from things like chemical spills, dangerous turbines, and natural disasters. According to Porfiri, "Schooling fish have a rich system of information sharing. They decide when to school based on a wide variety of factors, including vision and pressure cues from other fish. By studying these cues, we can learn how school members recognize — and follow — a leader." Apparently, groups of fish have shown various interaction patterns with the robot, including tracking, milling, and following, which indicates that a robotic member of the group can influence its behavior. If you want a demonstration, try to catch the professor and his students at the New York Aquarium where research will be ongoing through the academic year.

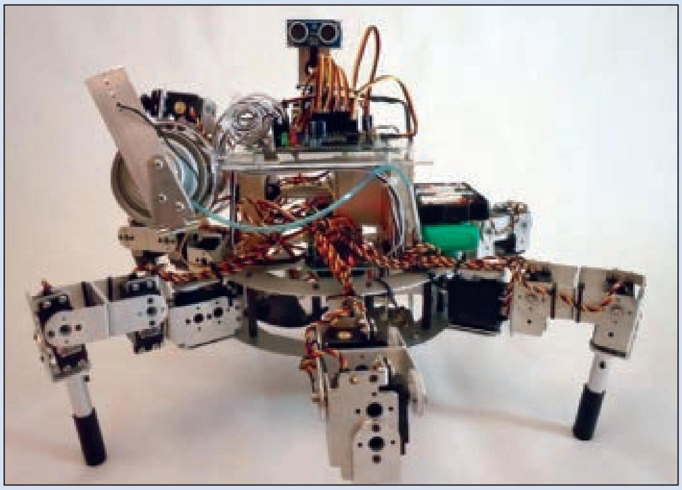

The second generation of robotic fish. (Courtesy of Polytechnic Institute of New York University.)


Robotic lander during strapdown testing. (Source: NASA/MSFC/David Higginbotham.)

Tests Begin on Robotic Lander

NASA has been working on a new generation of small robotic landers aimed at conducting research on airless celestial bodies such as the moon and near-earth asteroids. After only 17 months, the agency announced that the first prototype had been designed, built, and given its first tests at the renowned Redstone Test Center (**www.rtc.army.mil**) in Huntsville, AL. This initial test phase — known as "strapdown testing" — allows engineers to check out the integrated lander before moving to more complex free-flight tests. The program involved hot-fire tests to validate the propulsion system's response to guidance, navigation, and control algorithms and flight software before it undergoes autonomous free-flight tests. Because the lander won't be operating within an atmosphere, aero-braking and parachutes are not applicable, so autonomous landings are a bit tricky. According to NASA, the unit passed all tests, so we can expect the next phase — involving flight tests of up to 60 seconds — to begin this summer.

Robotic Farming with Swarm Technology

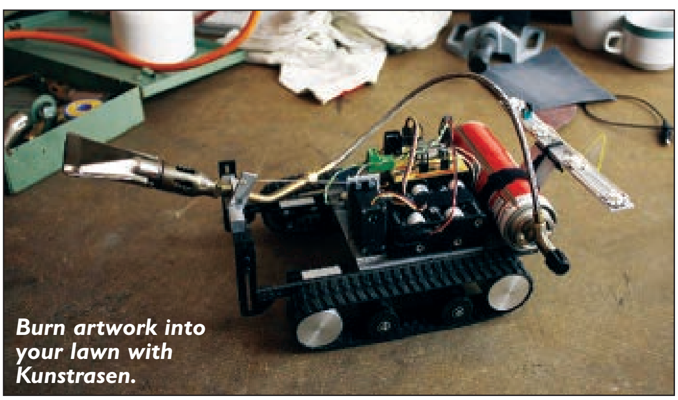


The Prospero Autonomous Micro Planter.

It may not pose much of a threat to John Deere, but the Prospero robotic farmer does offer a new approach to farming. According to its creator, David Dorhout, existing farm equipment has a basic design flaw in that it is centered around a human operator which has led to larger and more complex equipment to allow one farmer to cover as much as a thousand acres in a day. The problem is that such equipment tends to treat the entire farm as if soil, moisture, and nutrients are uniform which often is not the case. His alternative is the Prospero autonomous micro planter (AMP) which is designed to be one of a swarm of autonomous but interactive robots that — while planting — can adjust its procedures to match conditions in the field. The final goal is to develop one robot that can plant, tend, and harvest, autonomously moving from one phase to the next. The prototype is controlled by a Parallax Propeller chip mounted on a SchmartBoard. It can walk in any direction and avoid objects via ultrasonic pings. In operation, it can determine optimal spacing and depth, then dig a hole, plant a seed, cover the seed, and apply required fertilizers and herbicides. Several videos are posted on YouTube, so you can see it in action. A 60 page description of the AMP (including source code) is available from a blog on Trossen Robotics website. Just log onto **forums.trossenrobotics.com/showthread.php?t=4669** and scroll down to the PDF.

Scorched Earth Machine

Artist Sebastian Neitsch has created a range of robotic and kinetic devices, including a chandelier whose 12 lighted arms react to the movements of spectators, a kinetic garden of 36 moving carbon staves, a robotic cube that runs away from you, and others, most with no practical purpose. However, you have to like the possibilities presented by "Kunstrasen" (German for "artificial turf") — a small bot that crawls around on your lawn and burns designs into the grass using vector graphics and a flamethrower. Perfect for burning obscenities into your neighbor's turf or tormenting the greenskeeper at the local golf course. You can check them all out at **www.sebastianneitsch.de**. SV



Burn artwork into your lawn with Kunstrasen.

GEERHEAD

by David Geer
Contact the author at geercom@windstream.net

Robonaut 2: The First Robot On The International Space Station

Not "on a pony … named Wildfire", but on the Space Shuttle Discovery the Robonaut 2 rode beyond our atmosphere and up to the International Space Station (ISS). On February 24th at 4:53 pm EST, the Shuttle left the Kennedy Space Center with the first-ever humanoid robot in space which will be mounted inside the ISS for experimentation. There, astronauts will interact with the robot and aid it in developing skills that will some day prove useful in assisting future astronauts in their daily space-borne duties.

The chief roboticist at GM spoke with me about the historic shuttle event. "I attended the Discovery launch," says Marty Linn, principal engineer, robotics, GM, "I had never been to one. It was an amazing site. The weather was cooperative. We put the first humanoid robot in space. They moved the robot out to the shuttle in a box and transported it to the ISS in the module it will be housed in."

Though the robot is still under development, Robonaut 2 already hosts a broad platform of technologies that enable it to perform dexterous and manipulative tasks. Though its roots are 14 years old (NASA started Robonaut development in 1997), it has come a long way.

Robonaut 2 with a piece of hardware in hand.

Robotic Freedom (of Movement) in Outer Space

Today's Robonaut 2 consists of a torso, head, neck, and two arms, the two most dexterous hands ever mounted and functioning in robot history. Each joint across the robot's humanoid figure has a motor controller to actuate it based on commands from the brainstem.

While the robot's hands do not have all the same dexterity of human hands, each human-sized hand has 12 degrees of freedom (DOF) of movement for grasping and manipulating tools and objects.

The opposable thumb has four DOF for grasping and releasing, and moving in and out. Each middle and index finger has three DOF, and the ring and pinky fingers each have one DOF. With each finger gifted with five lbs of force, the robot can physically perform any task that its intelligence will learn and develop as a skill over the coming years.

"Six DOF sensors are mounted in the finger phalanges for tactile sensing capability," says Linn. There are four sensors that are strain gauges that resolve the six DOF forces that are applied on the phalanges. These are custom-built sensors based on strain gauge technology. "They measure the force applied by the fingers. We use them to resolve the six DOF forces based on information from the robot's hand contacting other objects," explains Linn.

Robonaut's 2′ 8″ arms are capable of seven DOF each, and have the power to hold up and control 20 lbs in any orientation. If Robonaut were to perform an Iron Cross on the rings a gymnast uses, its "wingspan" would stretch eight feet from end to end. Its neck has three DOF for turning so its head can look left and right, as well as up and down.

The head houses four light cameras: two for stereovision and two auxiliary cameras. "The robot uses two gigabit HD cameras in conjunction with the range finder for traditional machine vision types of automated tasks such as pose estimation, object identification, and tracking. It uses two low resolution analog cameras for teleoperation. Along with the tele-op immersion gear, the operator can control the robot motions and see what the robot sees," Linn commented.

R2 Intelligence

The torso — which connects to every other part of the robot — houses the robot's computer brain, called the brainstem or controller. The robot's brainstem is the computer chassis in the stomach of the robot that sends the commands that make Robonaut move. It takes commands and turns them into actions in the joints on the robot. The controller also monitors a number of different safety features in the robot. "NASA has rules for those kinds of things. The control system is triple redundant and safe," says Linn.

The engineers coded the brainstem of the robot in C++ and C#. There is a lot of sensor data processing software in

Getting a grip on Robonaut 2's awesome dexterous grasp in a laboratory setting.


Engineers work on Robonaut's arm.

the robot, as well as software to control forces and positions which the engineers have embedded in the individual joint electronics.

The robot is developing machine learning and other intelligence by capturing manually generated trajectories for reuse and recognizing simple object poses with the tactile sensors. The robot collects object position data through the machine vision system (the camera and range finder), and through the haptic sensing in the fingers and arms.

The robot can be automated to perform tasks on its own or — with the flip of a switch — the astronauts can change its mode in the brainstem taking it from automated operation to tele-operation as the command feed.

Experiments in Space

Robonaut 2 will take part in a large variety of experiments while it is mounted inside the space station over the coming years. The engineers have designed and will further develop and train the robot to work side by side with astronauts in space inside and outside the space station.

The robot will start its training with simple tasks as it interacts with a task board. "As R2 earns its stripes, astronauts and engineers will develop additional tasks for it centered around helping astronauts to perform work on the ISS," says Linn. Eventually, they will task the R2 with wiping down handrails and cleaning air filters. "Because the ISS is a giant bubble of air orbiting in space, astronauts spend a lot of time cleaning things. This is tedious and takes them away from their core duties. With the help of R2, they will someday be able to leave those tasks behind to work on more important things," explains Linn.

Though the robot is currently in its experimental phase, engineers intend for it to work safely with and around astronauts to lighten their burden. The robot must handle itself in the weightless environment. In initial experiments, the astronauts will be working with the robot doing rudimentary things like pushing buttons, flipping switches, and dealing with bags and compartments, plus teaching the astronauts how to program and work with the robot. The robot will manipulate space blankets, tether hooks, and power switches that have safety covers, as well as use drills and handling soft goods.

"There are tremendous synergies between GM and NASA with the overriding drive to ensure the safety of companies and workers. We have embedded this theme in everything we do with the robot. We envision robots and humans working together on different tasks in a safe manner," says Linn.

Practical Applications on Earth

General Motors personnel were co-located at the Johnson Space Center and worked side by side with NASA personnel from the first day of the collaboration, upgrading and outfitting R2. GM was involved in every aspect of the robot's development, build, and implementation.

GM plans to use Robonaut 2 technologies right here on earth. GM is looking at all of the R2 technologies including its sensors, mechanisms, and control systems for use in both GM plants and GM products. The focus will be on improved safety and quality. Though GM has not announced any plans for deployment as of this writing, GM sees many opportunities for real world applications of Robonaut's advanced engineering.

"The possibilities are endless," says Linn, "limited only by the imagination. We are exploring all kinds of applications for the safety and control systems in the robot, as well as its machine vision. Each technology has individual potential applications in the GM plants and products. As we build electric cars for example, such as the Chevy Volt, we may be able to endow it with attributes of autonomy similar to those exhibited in the DARPA Urban Challenge car."

The future may bring the evolution of cars that are smarter and won't allow drivers to do things that could cause harm. "We want to make the technology real and practical to the consumer. Autonomous vehicle operation has been

Resources

Follow Robonaut's Tweets at **http://twitter.com/AstroRobonaut**

Video demonstrating R2's dexterity **www.youtube.com/watch?v=vXdvVdKyzPM&feature=player_embedded#at=17**

showcased in the concept cars in urban commuter vehicles. Not only autonomous operation, but the improvement of traffic flow, sensing other vehicles to avoid collisions, and communicating with other vehicles and infrastructure to improve traffic congestion and curtail pollution," Linn comments.

Outfitting R2 with a space suit suitable for the ISS.

Conclusion

Robonaut is still early in its development. Who knows what it will evolve into in decades to come. **SV**

Photos and captions courtesy of GM.

ASK MR. ROBOTO

Our resident expert on all things robotic is merely an email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

by
Dennis Clark

It's May and I'm not ready for the winter to be over. Springtime means that I need to shift into high gear to get robots ready for school demos and upcoming summer and fall robot competitions. Sigh, so many projects and so little time!

Enough whining. Let's get to those projects and questions!

Q. I have a biped robot that I'm trying to get to walk. My problem is that I have to pick it up while I'm testing a stepping gait or it might shred itself with a movement that is too fast. I know that some kits come with a stand of some kind to help with this. How should I make one?

— Pete S., Denver, CO

A. Pete, not all kits come with a stand. I have a few bipeds and only ONE of them came with a stand to test movement on. Your best and cheapest route is good ol' PVC pipe from your local home improvement store. I used 1/2" Schedule 40 PVC. To build this stand, you will need five corner joints and two T joints, along with a single 10 foot length of 1/2" PVC pipe. I got the store to cut it in half to fit in my car. The total cost was under $7 for all the pipe. The most expensive part was the PVC primer and cement; get the smallest and cheapest ones you can, it isn't that important as long as the joints stick.

Construction is very simple. Normally, I don't bother to glue my PVC construction creations, but in this case, we're hanging a fairly heavy robot and the pipe joints will slip. You will only need to glue the joints that hold the vertical post (the joints marked in purple in **Figure 1**).

Figure 1. Biped rack.

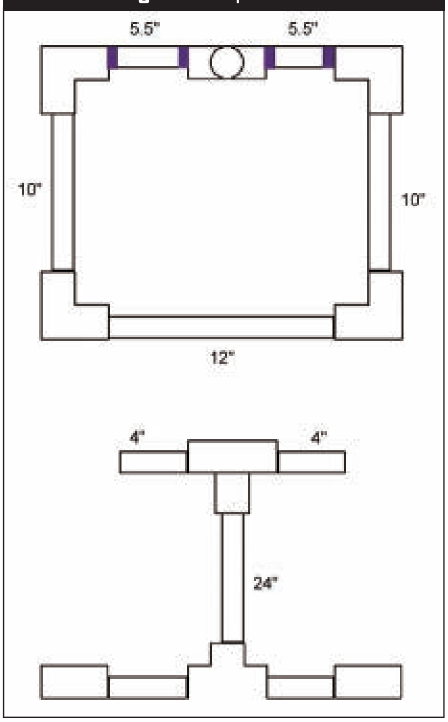
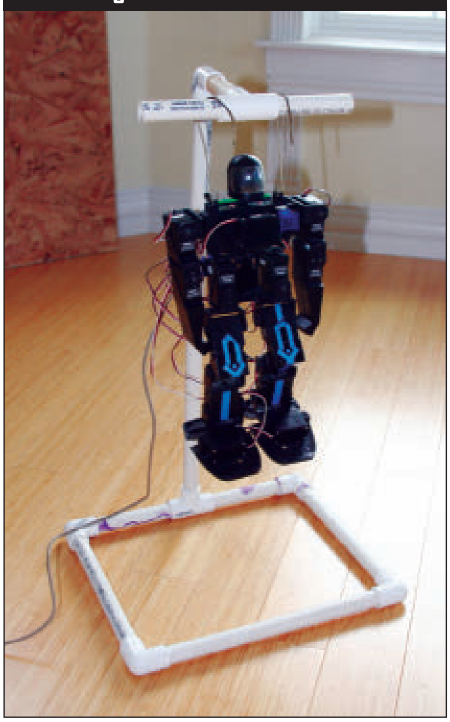


Figure 2. The rack in use.



I've marked my chosen dimensions in **Figure 1**. They aren't critical; just make everything match up and be large enough to be stable. Make sure your vertical pipe is long enough! My first attempt wasn't. **Figure 2** shows my RoboPhilo hanging; it has holes on its chassis just for this purpose. Neither my RoboBuilder nor Bioloid have those holes; I had to jury-rig them. I use wire clothes hanger wire for the actual hooks. Have fun!

Q. I have been researching robot motion — mostly what type of wheel mechanism to use. I keep running into references to encoders that are intended to sense how fast/far the robot is moving.

Why not take the sensor/circuitry from an infrared mouse and monitor it with a low cost processor (BASIC Stamp)

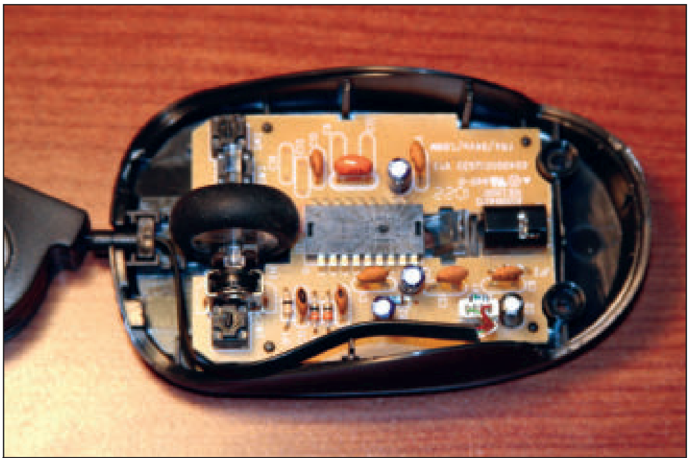


Figure 3. USB only single chip mouse.

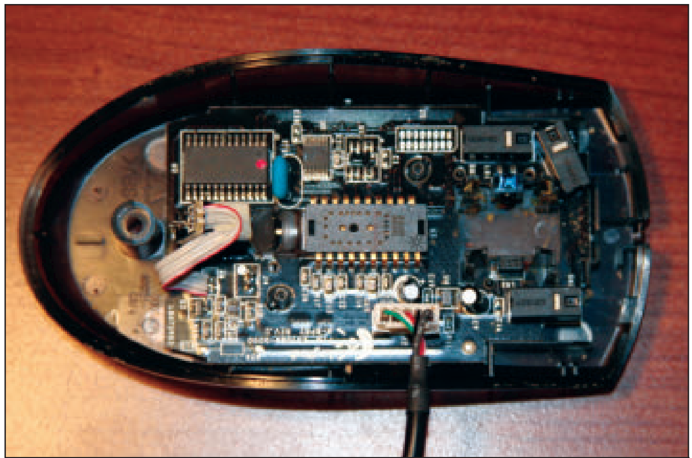


Figure 4. USB/PS2 mouse.

that sends status updates to the main processor (Arduino Uno)?

I am a relative beginner to robots but do have a lot of programming experience, and some with circuitry. Am I just being naive with this question? It seems like such a simple solution to know where your robot is. I would like it to be able to keep an internal map of its surroundings.

I really enjoy reading your column each month. Thanks for putting the effort into it to keep it fresh and interesting.

— Bob Patterson
Portland, OR

A. Bob, you are not being naïve at all. However, while this sounds simple and my searches found websites that ALSO looked simple, this isn't all that easy to implement when compared to driving a DC motor or a servo, that is. I would recommend the Arduino (for instance) for the optical mouse monitoring because it is much faster and cheaper than the Stamp would be. I'd been having the same thoughts about optical mice and robot tracking, so I figured that I would try my hand at it myself. The first hurdle was just *finding* an optical mouse that I could hack to do the project!

Most of the web searches I did talked about using a PS2 mouse since it was basically synchronous serial communications based. This is easy with an embedded microcontroller (more later), so I went in search of said item. Hmm. No one makes PS2 mice anymore. But I did learn that many USB mice were made that with the help of an adapter, could also be PS2. Armed with that knowledge, I hit the stores. My first find (under $10 of course) was a Dynex mouse (**Figure 3**). The single chip solution told me two things: it was not hackable at the chip level (USB capable) and it was also not a USB/PS2 switchable mouse. I then went to a local shop that recycles computer stuff for cheap. I got an older Logitech optical mouse that looked promising (**Figure 4**) since I read it was PS2 compatible. Since the PS2 protocol is pretty well known and documented, I decided to experiment with this one.

The optical chip in the MX300 is the Agilent ADNS2020. Not only is this chip not produced any more, you can't even get documentation on it. I tried looking at the ADNS2030 chip documentation but they don't have the same pin-out, so that wasn't going to fly. Intrigued at using these devices, I went looking for optical mouse chips at my favorite distributors (like Digi-Key and Mouser) and found some. For instance, at Digi-Key you can find the Avago ADNS-5050 eight-pin optical sensor and mate it with the ADNS-5100 lens for less than $2.50. Why hack when they are so cheap in quantity one orders? Well, you hack when you have a mouse on your desk and don't want to wait three days! I plan to get a couple of these to play with, but today I hacked.

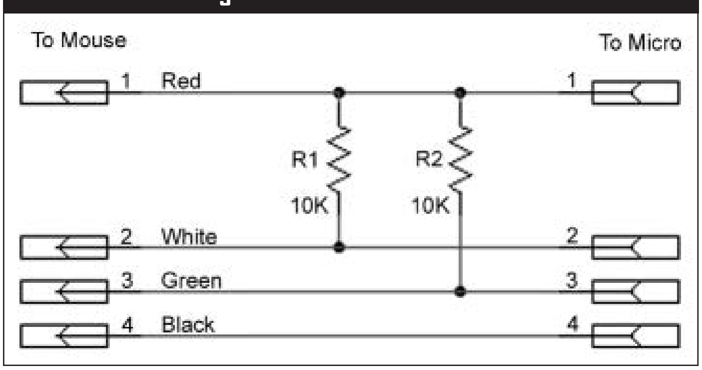
Another quick trip to Google found me the pin-out of the mouse and the wire colors which appear to be standardized in the mouse world; see **Figure 5** for the wires to use and their functionality.

I'm not bothering with the USB connector; I just hacked that off and made my own connectors for my

Function	USB Pin #	USB Color	PS2 Pin#	PS2 Color
5V power	1	Red	4	Red
SDATA	2	White	1	White
SCLK	3	Green	5	Brown
Ground	4	Black	3	Black

Figure 5. USB/PS2 wiring.

Figure 6. Mouse connections.



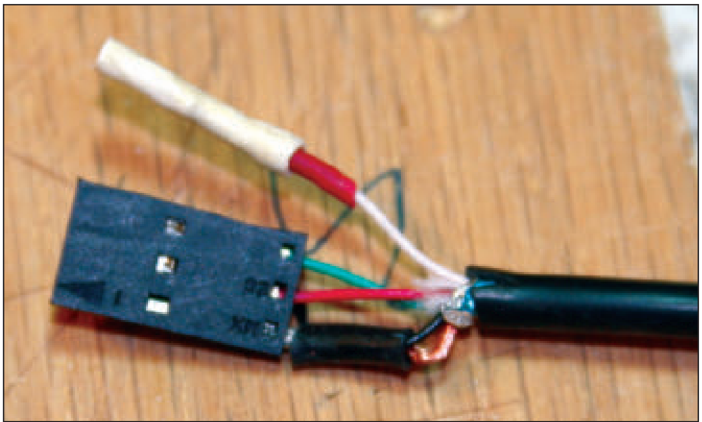


Figure 8. PS2 robot connector.

Command Number (Name)	Explanation and Use
0xFF (Reset)	The mouse responds with "acknowledge" (0xFA), then enters reset mode.
0xF6 (Set Defaults)	The mouse responds with "acknowledge" (0xFA), then sets: sampling rate = 100; resolution = four counts/mm; scaling = 1:1; data reporting = disabled. The mouse then enters stream mode.
0xF5 (Disable Reporting)	The mouse responds with "acknowledge" (0xFA), then disables data reporting and resets movement counters. Disabled stream mode functions the same as remote mode.
0xF4 (Enable Reporting)	The mouse responds with "acknowledge" (0xFA), then enables data reporting and resets movement counters. This command may be issued while the mouse is in remote mode, but it will only affect data reporting in stream mode.
0xF3 (Set Sample Rate)	The mouse responds with "acknowledge" (0xFA), then reads one more byte from the host. The mouse saves this byte as the new sample rate. After receiving the sample rate, the mouse again responds with "acknowledge" (0xFA) and resets movement counters. Valid sample rates are 10, 20, 40, 60, 80, 100, and 200 samples/sec.
0xF2 (Get Device ID)	The mouse responds with "acknowledge" (0xFA), followed by its device ID (0x00 for the standard PS/2 mouse). The mouse should reset movement counters.
0xF0 (Set Remote Mode)	The mouse responds with "acknowledge" (0xFA), then resets movement counters and enters remote mode.
0xEE (Set Wrap Mode)	The mouse responds with "acknowledge" (0xFA), then resets its movement counters and enters wrap mode.
0xEC (Reset Wrap Mode)	The mouse responds with "acknowledge" (0xFA), then resets movement counters and enters the mode it was in prior to wrap mode (stream mode or remote mode).
0xEB (Read Data)	The mouse responds with "acknowledge" (0xFA), then sends a movement data packet. This is the only way to read data in remote mode. After the data packet has successfully been sent, the mouse resets movement counters.
0xEA (Set Stream Mode)	The mouse responds with "acknowledge" (0xFA), then resets its movement counters and enters stream mode.
0xE9 (Status Request)	The mouse responds with "acknowledge" (0xFA), then sends the following three-byte status packet (then resets its movement counters).

Listing 1. Relevant PS2 commands.

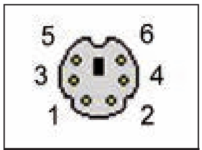


Figure 7. PS2 cable connector.

experiments. If you actually wanted to use the mouse as a pointer device, you would no doubt use proper mini-DIN connectors (but we're looking at hacking this onto a robot, so not interested). **Figure 7** shows the pin-out of the PS2 connector as seen looking at the end of the mouse cable.

Interfacing the PS2 Mouse Hardware to an ATMEGA168

To get the mouse to act like a PS2 device, the Data line must be pulled high; a 10K resistor works well. This is both a signal to the USB/PS2 controller chip to use PS2 (this line would normally be low in USB mode), but it's also needed because the Data line is used to both transmit and receive and the pull-up will keep the line from floating when the master and slave devices are switching between input and output modes. See **Figure 6** for schematic details.

There are four wires needed for the PS2 serial connection so obviously this isn't very convenient for our typical three-pin proto board or robot board connections. Just make a standard three-pin connector (I used 5V, ground, and the clock line) and make the fourth wire its own connector as shown in **Figure 8**.

Talking to the PS2 Mouse to Find Movement Values (the Protocol)

Now, we need to write some code to talk to our new motion encoder. There are several websites out there that have the PS2 mouse protocol defined. They all agree on the command set, so I picked one at random and tried it out. **Listing 1** has the commands we need to get our X and Y coordinate changes. The clever person could perhaps find a way to hack in binary proximity sensors to the mouse button switches and get even more sensor values from our purloined mouse circuit board. (I'll leave that as an exercise for the reader.) I haven't put all of the commands in this list (only the useful ones) and won't even use all of these for getting the mouse to respond with basic movement data. There are extended command sets for later, to use with more precise mice that can be configured. Let's see this basic set work before we get fancy.

The standard way a mouse is used is to set it to Stream mode and the mouse will send a data packet (we'll see that in a moment) whenever it senses movement. An alternative way to get data from the mouse is to use Remote mode. In Remote mode, the master controller must send a Read Data command to get a movement packet. The movement packet is shown in **Listing 2**.

The movement packet sends a nine-bit 2's complement number for the X and Y movement values; a 2's complement means that the Most Significant Bit determines the sign of the value. A '1' in the X or Y sign bit means negative (backwards) movement with respect to the typical orientation of the mouse. Beware then of how you mount

the mouse so that you make sense of this direction. If movement since the last data reported is greater than ± 255, then the X or Y overflow bit will be set.

The above information is the easy part. Now comes the harder part. The mouse is always the generator of the clock; sometimes the master (our microcontroller) sends data, sometimes the mouse does. The master still controls this, however, by pulling CLK or DATA low, depending on the action required. We need to play games with setting the CLK and DATA lines as outputs and inputs at certain times and at very careful timing rates. Another problem is that the PS2 mouse serial data line is 11 bits, not eight or 16 like we expect for SPI communications. Therefore, we need to bit bang our mouse port rather than use any of our processor's hardware synchronous serial modules.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	1	Middle button	Left button	Right button
Byte 2	X movement							
Byte 3	Y movement							

Listing 2. PS2 Movement Packet.

Ready? Let's learn the hardware protocol now.

The basic configuration of the PS2 bus is that the CLK and DATA lines must be Wire-OR'd. This means that a logic low always wins. If one side pulls the line high and another device pulls the line low, then the line will go low. This must be accomplished without any device on the signal line fighting the other. The way that this is usually done is with what is called open collector or open drain logic. The signal line will have a pull-up resistor on the line so that when no one is driving the line low, the signal is pulled high. Note the circuit in **Figure 6**. We use pull-ups on the CLK and DATA lines for this reason.

Because it is annoying to hack resistors onto a controller board or onto a connector, we have a shortcut we can use on a microcontroller. Most of them have the ability to use an internal pull-up. Both the PIC and ATMEGA microcontrollers can do this. We too will do this to save on parts and complication. The secret to implementing the Wire-OR'd signal lines is to change the I/O line into an input when we want the line to go high, and to an output with a logic '0' written to it to pull the signal low. This is as easy as writing a data bit value to the I/O port. We write the 1 or 0 to the Data Direction Register for that I/O line. I'll show you how in code that follows.

Sending a Command to the PS2 Mouse

The PS2 interface uses a synchronous serial protocol that is 11 bits long. Refer to **Figure 9** for this discussion. Data is bracketed with a start bit (always 0) and a stop bit (always 1). Eight data bits and a parity bit are sent with the Least Significant bit (bit 0) first and the parity bit last. The PS2 protocol uses odd parity which means that we set or clear the parity bit to have an odd number of 1 bits in the data stream (not counting the parity, stop, or start bits).

The master changes the data output when the CLK line is low; the PS2 mouse will read the DATA line when the CLK line is high. To start a transmission to the PS2 mouse, the master first pulls the CLK line low for 100 microseconds, then pulls the DATA line low, and finally releases the CLK line. All of this sends a Request To Send to the PS2 mouse. The master controller then waits for the PS2 mouse to pull the CLK line low and starts the transmission sequence as shown. When the master controller releases the DATA line, the PS2 mouse will generate one more clock and pull the DATA line low as an Acknowledge (ACK for short) when the CLK line is high (unlike the rest of the data bits from the master controller). Because the clock rate is typically between 10 and 16 kHz, I'm going to use an interrupt line to set/clear DATA bits on CLK transitions. This is far more efficient in processing time than polling the CLK line.

Getting Data from the PS2 Mouse

Getting data from the PS2 mouse is very similar to sending a command, only the consumer changes; the CLK signal is still from the mouse. Refer to **Figure 10** for this discussion. The data exchange is again 11 bits with a start bit (always 0), eight data bits, a parity bit, and the stop bit.

The device starts the transmission by pulling the CLK line low and changing the data when the CLK line is high. The master controller reads the data bit when the CLK line is low — the opposite as with a master-to-device exchange.

Whew! My time is up for this month. Next month, we'll write the code and implement the device as a robot movement sensor to see how well it works.

Until next month, keep building those robots and as usual, if you have a question, please send me an email to roboto@servomagazine.com and I'll be happy to try to answer it. SV

Figure 9. Controller sending data to mouse.

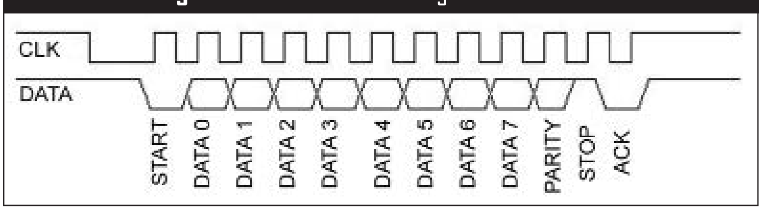
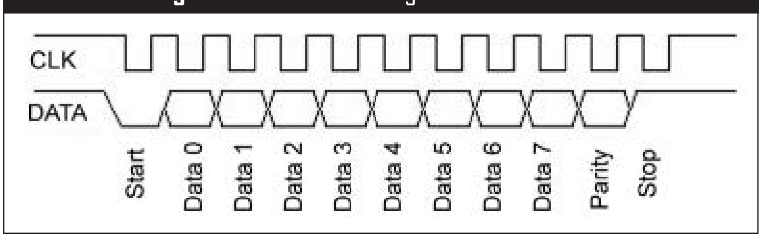


Figure 10. Device sending data to controller.



EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

MAY

7 RoboFest
Lawrence Technological University, Southfield, MI
Several events include a cooperative competition in which two autonomous robots must work together. Also a RoboFashion show, Mini Urban Challenge, Fire-Fighting, VEX, and a robot exhibition.
http://robofest.net

9-13 FIRA Robot World Cup
Dubai, UAE
The whole range of autonomous soccer robots from tiny Kheperas to bipedal humanoids will compete in matches.
www.fira.net

9-13 ICRA Robot Challenge
Shanghai, China
This year, there are three events: the Micro-robot Challenge; the Solutions in Perception Challenge; the Modular and Reconfigurable Robot Challenge; and the Virtual Manufacturing Automation Challenge.
www.icra2011.org/show.asp?id=40

14 DPRG Roborama 2011.a
Dallas, TX
The first of the Dallas Personal Robotics Group's two annual competitions includes some traditional events, as well as new events for 2011. Indoor events include the VEX Unlimited Open, Square Dance, Line Following, and Table Top. The outdoor event this year is RoboColumbus.
www.dprg.org

14 Western Canadian Robot Games
Calgary Aerospace Museum, Calgary, Alberta, Canada
There's always a lot of cool stuff going on at the WCRG. Past events have included Sumo, Mini-Sumo, a walking robot triatholon, robot art, atomic hockey, fire-fighting events, and, of course BEAM robots.
www.robotgames.com

20 NATCAR
UC Davis Campus, Davis, CA
A high speed variation on line following in which the line is a wire taped to a black floor with white tape. The wire produces a 75 kHz magnetic field, so either optical or magnetic sensors may be used.
www.ece.ucdavis.edu/natcar

20 SPURT (School Projects Using Robot Techniques)
CMU, Pittsburgh, PA
Student-built 20 cm wide autonomous robots race on the SPURT track.
http://spurt.uni-rostock.de

20-21 Swiss Eurobot
Yverdon-les-Bains, Switzerland
This is a regional for the main Eurobot contest which is coming up in June this year. This year's contest is called Chess'up! and involves autonomous robots moving chess pawns and stacking them up to gain points.
www.swisseurobot.ch

23-26 NASA RASCAL Robo-Ops
Johnson Space Center, Houston TX
University teams compete to build planetary rovers that must perform a series of competitive simulated exploration tasks in the Johnson Space Center's Rock Yard. The robots use a combination of autonomous obstacle avoidance and teleoperation from a mission control center, where the operators have only data provided to them through the robot's sensors.
www.nianet.org/RASCAL/RoboOps/index.aspx

NEW PRODUCTS

ACCESSORIES

Aluminum Transmitter Case for the Aurora 9 Radio System

Now you can bring your radio and accessories everywhere you go with the new transmitter case available from HiTec. This new transmitter case is solidly constructed of aluminum for the ultimate in durability and style. The exterior has pre-cut foam padding for maximum cushioning and protection. Your transmitter fits snuggly in the center section while all accessories from receivers to servos can be arranged in the surrounding compartments. Optimally designed for the Aurora 9, this sleek carrying case will store all other Hitec radio systems as well. When you are on the go, it's easier to keep your radio safe. Stock number is 55445.

For further information, please contact:

Hitec

12115 Paine St.
Poway, CA 92064
858•748•6948 Fax: **858•748•1767**
Website: **www.hitecrcd.com**

MODULES

Wixel Programmable USB Wireless Modules

Pololu announces the release of the Wixel, a general-purpose programmable module with integrated full speed USB and a 2.4 GHz radio. Users can load their own custom programs or pre-compiled, open-source apps to enable wireless control of a robot, create a wireless microcontroller programmer, establish a wireless sensor network, and more. No programming experience is required to get started with the Wixel. Users can simply download an app from the Web and upload it to the Wixel using its built-in USB bootloader and Pololu's free configuration software (no external programmer is required). For example, the Wireless Serial App can be used to turn a pair of Wixels into a wireless USB/TTL serial link for communication between two microcontrollers or between a PC and a microcontroller.

The Wixel is based on the versatile CC2511F32 microcontroller from Texas Instruments which has an integrated radio transceiver, 32 KB of program memory, 4 KB of RAM, and a USB interface. The Wixel makes a total of 15 general-purpose I/O lines available, including six analog inputs, and the 0.1" pin spacing makes it easy to use with breadboards and perfboards. For those who want to write custom applications in C, the Wixel SDK provides open-source development tools and libraries, and Pololu's selection of apps can serve as starting points for custom programs. Individual Wixels (item #1337) are

available for $19.95 each.

For further information, please contact:

Pololu Corporation

3095 E. Patrick Ln. #12
Las Vegas, NV 89120
Tel: **877•7•POLOLU** or **702•262•6648**
Fax: **702•262•6894**
Email: **www@pololu.com**
Website: **www.pololu.com**

ADAPTERS

GSPA Series Prototyping Adapters

Global Specialties announces the release of a series of five new high-end prototyping adapters. The GSPA series includes a variety of PLCC and SOP to DIP adapter packages which are built to provide maximum surface contact for years of reliable use. The GSPA series adapters are perfect for education, prototyping, production, and R&D, where high reliability component testing and multiple-use applications are required. Each model is sold separately and is compatible with all Global Specialties electronic trainers and solderless breadboarding systems.

For full product specifications for these new high-end prototyping adapters, as well as the complete line of other Global Specialties Prototyping and Test & Measurement products, visit their newly updated website listed below.

For further information, please contact:

Global Specialities, LLC

22820 Savi Ranch Parkway
Yorba Linda, CA 92887
Call Frank Menichello: **800•572•1028**
Website: **www.globalspecialties.com**

bots IN BRIEF

FOR THE BIRDS

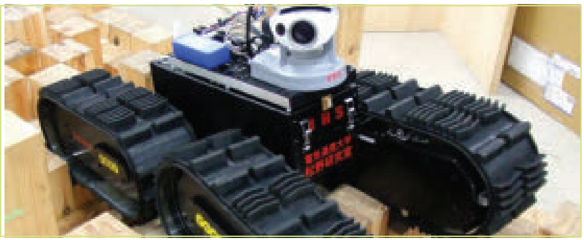
Festo's new Smart Bird can pretty much pass for a seagull. Made of light material (like carbon fiber) and sensitive control electronics, the mass is about 0.4 kilograms (a third less weight than an iPad). The wings move up and down, and also twist at angles like real ones. The bird sends data back to its operator so that they can adjust for optimum flight.

SmartBird is an ultralight but powerful flight model with excellent aerodynamic qualities and extreme agility. With SmartBird, Festo has succeeded in deciphering the flight of birds. This bionic technology-bearer — which is inspired by the herring gull — can start, fly, and land autonomously with no additional drive mechanism. Its wings not only beat up and down, but also twist at specific angles. This is made possible by an active articulated torsional drive which — in conjunction with a complex control system — makes for unprecedented efficiency in flight operation. Festo has thus succeeded for the first time in realizing an energy-efficient technical adaptation of the natural model.

PAGING FLORENCE NIGHTINGALE

A team from Purdue has turned a 59 lb Fanuc arm into a robotic scrub nurse. The team has spent the past five years creating code and teaching it to use natural language and gestures so that it can assist doctors. So far, the bot can handle scalpels, retractors, scissors, hemostats, and forceps by using a series of hand gestures. The robonurse should be available to hospitals in another five years.

Surgeons routinely need to review medical images and records during surgery, but stepping away from the operating table and touching a keyboard and mouse can delay the surgery and increase the risk of spreading infection-causing bacteria. The new approach is a system that uses a camera and specialized algorithms to recognize hand gestures as commands to instruct a computer or robot for these kinds of tasks.



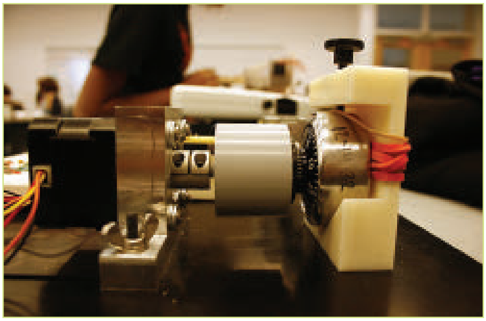
TO THE RESCUE

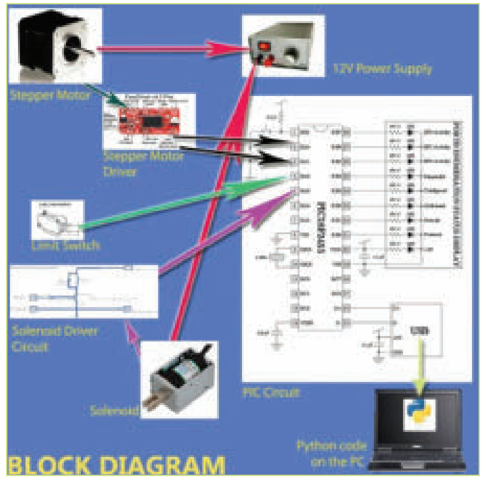
Japan's rescue teams were just not enough after the razing tsunami and massive earthquake that brought destruction to many coastline villages. The manned teams used searchdogs and performed basic first aid operations such as searching for survivors and helping them medically. But what about the survivors that were below a big pile of debris caused by a collapsed building? This is where multiple robot-aided teams came to the rescue. The first team was led by Prof. Eiji Koyanagi from Chiba Institute of Technolog; the second led by Prof. Fumitoshi Matsuno from Kyoto University, who is vice president of the International Rescue System Institute. Two other teams based in Tokyo and Sendai waited on standby.

MONI A MANO

Japan sent Monirobo — a five foot, 1,300 pound rescue bot — into the crippled Fukushima nuclear plant to measure radiation levels, humidity, and temperature. The bot runs on tank treads to easily maneuver over debris and sends 3D videos to its operator; although it can only travel about 3/4 mile away from it. Even though the country is said to have about 50 robots available for rescue, Moni was the first to get to work.

TOP PICK(ER)





Students at the Franklin W. Olin College of Engineering in MA came up with a robot that can pick a lock. Designed to break Master Locks, the user puts in as much of the combination possible (or none of it) and LockCracker does the rest. It also displays the number.

This flexible electro-mechanical system can be modified to fit any turn-dial lock, as well.

EN-V OF THE ROAD

GM has been busily working on the EN-V — a battery-run vehicle that does its own thing if you are too tired (or drunk) to run it yourself. The car has a wireless antenna and GPS sensors that connect with a network to become autonomous. Chat it up with other EN-V owners or push a button and it parks itself (six of them can fit in a standard space).

Not only can the vehicle run itself, it is in contact with others so it will not collide with them. Of course, the EN-V doesn't chat with current cars, so they will have to have their own roadways.

KASPAR THE FRIENDLY BOT

KASPAR (Kinetics Synchronization in Personal Assistance Robot) is used with kids in the UK that have autism. He talks, laughs when touched, and cries "Ouch" when someone slaps at him. Parents report that their kids are more affectionate with humans after playing with the bot.

KASPAR is a child-sized humanoid robot developed by the Adaptive Systems Research Group at the University of Hertfordshire. KASPAR is being used to study human-robot interaction as part of the European RobotCub Project which aims to build an open-source robot platform for cognitive development research. The Adaptive Systems Research Group is investigating the use of gestures, expressions, synchronization, and imitation. In addition, the robot may be used for developmental studies and interactive games.

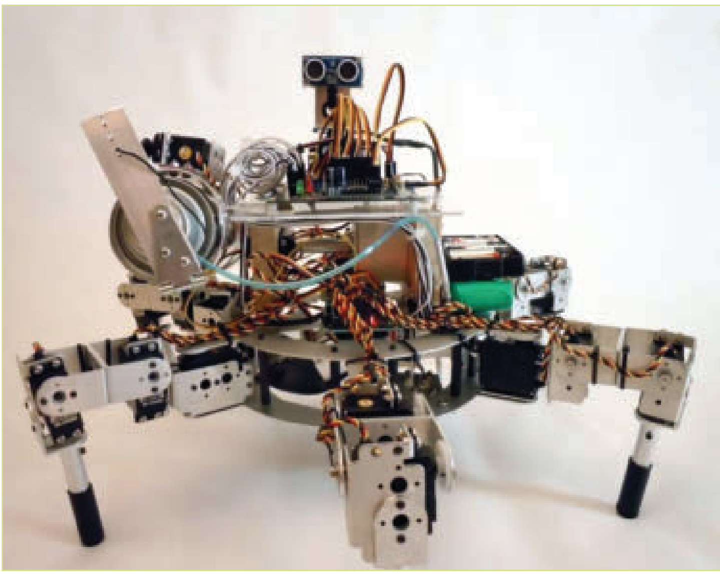
This family of robots has been used in the past in the Aurora project which investigated the possible use of robotic systems as therapeutic or educational tools to encourage social interaction skills in children with autism. They are also currently being used as part of the European FP6 IROMEC project and the European FP7 ROBOSKIN project. The IROMEC project acknowledges the important role of play in child development and targets children who are prevented from playing; either due to cognitive, developmental, or physical impairments which affect their playing skills. This project investigates how robotic toys can empower kids to discover the range of play styles from solitary to social and cooperative play.

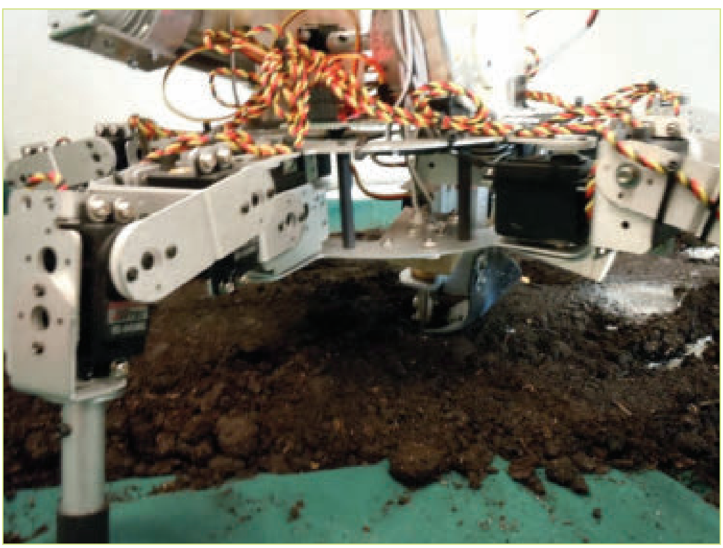
The ROBOSKIN project will develop new skin sensor technologies that can provide tactile feedback from large areas of the robot body. A team at UH will help to develop cognitive mechanisms that use this tactile feedback to improve human-robot interaction capabilities, and implement it in the problem domain of robot-assisted play for children with autism.

Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

BEE IN CONTROL

BeeWi's Mini Cooper doesn't just run by RC, it runs via your smartphone. Compatible with Android and some Nokia Symbian smartphones, the Mini Cooper is controlled by orientation sensor and touchscreen buttons. The 179 x 98 x 74 mm BBZ201-A0 zips along on three AA batteries for up to three hours before needing a recharge.





DIGGIN' IT

The five-legged Prospero is a clever bot that is being used as an autonomous micro planter (AMP). It joins with others to make a swarm that can plant seeds, tend them, and harvest the finished product. Controlled by a Parallax Propeller chip mounted on a SchmartBoard, it can walk autonomously while avoiding objects. Prospero can also communicate with other robots and can carry fertilizer and herbicides in its belly.

Boasting five legs, this machine is capable of doing all it requires to grow a plant. It works autonomously, navigating in any direction and avoiding obstacles. A sensor located right underneath its body lets it know where seeds have been deployed, and if it thinks that a particular spot underneath it needs some seeds, it will dig a hole, plant the seed, and spray the ground in white to indicate success.

Prospero is the working prototype of an AMP that uses a combination of swarm and game theory, and is the first of four steps. It is meant to be deployed as a group or swarm. The other three steps involve autonomous robots that tend the crops, harvest them, and finally one robot that can plant, tend, and harvest — autonomously transitioning from one phase to another.

Prospero checks to see if a seed has been planted and if not, plants one and then marks it with a biodegradable paint spot. It then moves on to another spot, all the while signaling the other hexapods its status.

The signals to the other hexapods are sent by infrared signals. This is currently shown via LEDs. Green LEDs signal the other robots to come closer as seeds need to be planted. Red LEDs signal the other robots to stay away as seeds have already been dispersed.

IT AIN'T HEAVY, IT'S MY AVATAR

Looks like there's some competition in the "wearable robot" category. Unveiled publicly at Interaction 2011 in March, TEROOS was built using commercially available servos and 3D printed parts designed by researchers at Keio University's Anzai-Imai Lab and ATR's IRC lab. It takes telecommunication to the streets by slinging it over your shoulder any time you go out. A simple PC interface displays a live video feed (through Skype) and sends commands to the robot through a cellphone connection that communicates instructions over Bluetooth. The operator can look around by tilting and panning the robot's head, converse through the built-in microphone and speaker, and even make simple expressions like blinking.

"With video chat, people can communicate face to face over a long distance. But the problem is that going out or shopping together can prove difficult. With this system, the person who's going out carries an avatar on their shoulder, so the other person can operate the avatar to look around freely. This can provide an experience just like going shopping or having a date with the other person."

Hmmm ... could be a new way to monitor the kids or bond with your mother-in-law.

IS IT REAL OR ...

This is the latest iteration of the Geminoid series of ultra-realistic androids from Kokoro and Hiroshi Ishiguro. Specifically, this is Geminoid DK which was constructed to look exactly like Associate Professor Henrik Scharfe of Aalborg University in Denmark.

Just like with the other Geminoid robots, all of the movements and expressions of Geminoid DK are remote controlled by an operator with a computer who uses a motion-capture system that tracks facial expressions and head movements. Turn your head and the Geminoid does the same; move your mouth and the android follows suit.

COMBAT ZONE

Featured This Month:

www.servomagazine.com/index.php?/magazine/article/may2011_CombatZone

BUILD REPORT

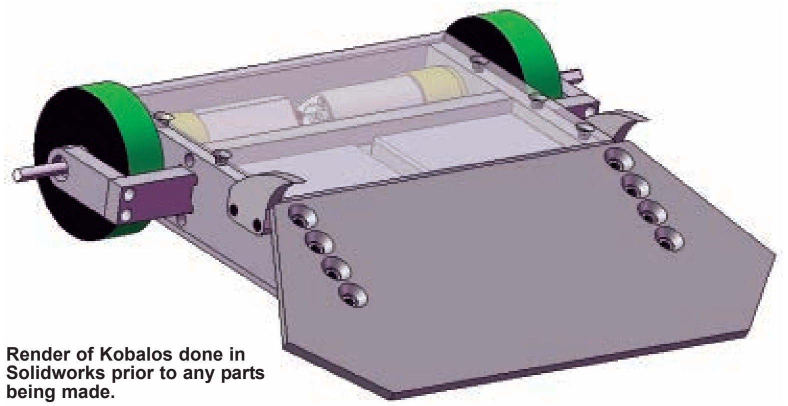
Kobalos – Antweight Wedge/Rammer

● by Mike Jeffries

Kobalos is a miniaturization of my 12 lb robot, Apollyon. The basic idea was to build something that would be as close to indestructible as possible. At the time of designing Kobalos, it had been almost five years since I had built my last 1 lb robot.

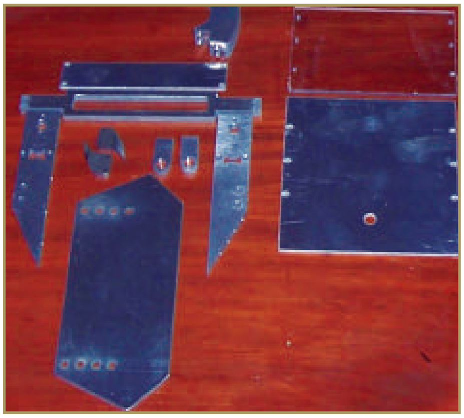
The first step in the design process was to figure out what the modern components of choice were. After some investigation, I settled upon FingerTech Gold Spark motors at a 20:1 ratio for drive and paired them with the FingerTech TinyESCs. One 250 mAh Lipoly pack later and the majority of the new guts were figured out.

The chassis design was meant to be very simple. The four main pieces of the chassis provide 95% of the strength of the robot and were all made via waterjet cutting. The side rails and internal rail are 3/16" thick 6061 aluminum and are designed to allow the side rails to slide onto the internal rail. The sides are secured with 4-40 screws.

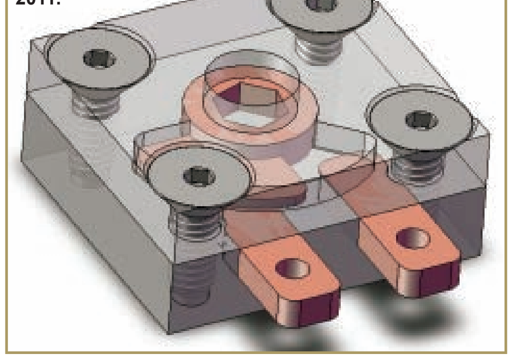


Render of Kobalos done in Solidworks prior to any parts being made.

Parts fresh from TeamWhyachi.com and ready for assembly.



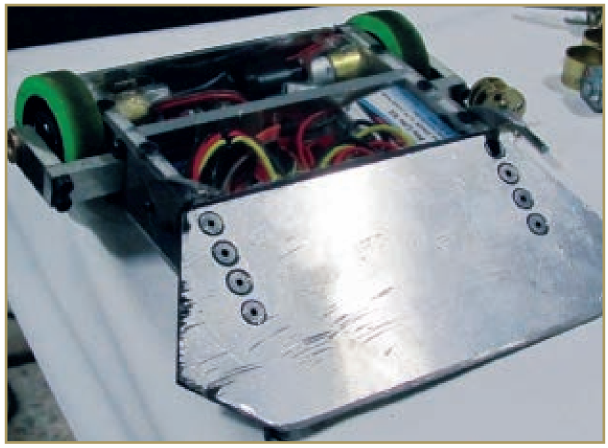
Power switch design used in Kobalos at Motorama 2011.



Once the side rails are attached, the front wedge is bolted on. With 3/32" thick 4130 steel for the wedge, it is a major structural component of the robot and a decent portion of its overall weight.

The next portion of the build was designing a custom power switch for Kobalos. More often than not, the power switch in the insect classes is nothing more than a jumper that is plugged in and shoved into an open hole on the robot's armor. I set out to design a better, safer power switch. The end result was a very small, very expensive switch weighing less than four grams and costing far too much to produce commercially. The switch worked as designed and has yet to be turned off unintentionally.

The wiring and building of Kobalos took less than half a day due to the simple nature of the robot and the use of waterjet cutting for the majority of the chassis components. Four small neodymium magnets were added to take advantage of the steel floored arena at Motorama 2011. There are a few minor bugs to be worked out, but overall, Kobalos performed very well at Motorama. SV



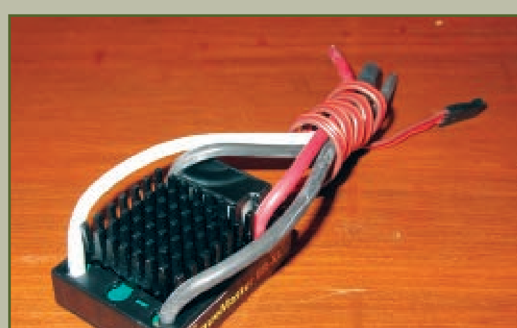
Kobalos after two fights and a rumble at Motorama 2011.

PARTS IS PARTS: Holmes Hobby TorqueMaster BR-XL Speed Controller

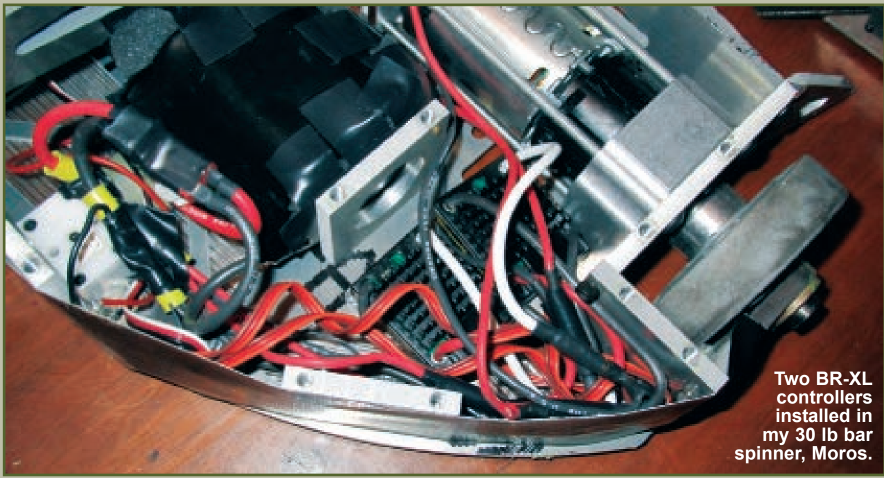
● by Mike Jeffries

When designing the latest versions of my 12 lb wedge and 30 lb bar spinner, I decided to look for an alternative to my relatively large and heavy Victor 883 speed controllers. While searching, I came across the Holmes Hobby TorqueMaster BR-XL and was impressed by the listed specifications. The controller was listed as handling 80A continuous on up to a 6s lipoly pack, while only weighing 38 grams. I decided to take a shot and ordered a set to test with.

The original test set was placed in my second generation bar spinner, Mr. Self Destruct. During initial



BR-XL controller freshly removed from packaging and ready for installation.



Two BR-XL controllers installed in my 30 lb bar spinner, Moros.

testing, they seemed to perform flawlessly. During a full system test, Mr. Self Destruct self-destructed, and the controllers were tossed around viciously. When inspecting the controllers, I found that the PWM cables had been sheared due to the forces they experienced. I trimmed the PWM cables and carefully soldered the new connections in place. I was impressed with how well the controllers handled the abuse they had seen and ordered several more BR-XLs.

In preparation for Motorama 2011, the BR-XLs were installed in Apollyon and Moros. Apollyon got a fresh set of controllers and Moros got the set that had been tossed around when its predecessor exploded. I continued my testing with the BR-XL controllers; and found no issues. They were easily calibrated to both my DX6 and DX3 transmitters and provided very quick response to the input they were given.

At Motorama, I did lose two of the BR-XL controllers, not due to electrical failure, but because they were ejected from Moros and were hit by Steel Shadow — a 30 lb shell spinner. After picking up the pieces, I noticed that the FETs were still attached to the heatsinks. Besides the forcible removal of the FETs, the boards appeared to be undamaged. It took less than 20 minutes to swap in a new set of BR-XL controllers and get them properly calibrated. Outside of severe mechanical damage, I have yet to lose a single BR-XL controller.

In both robots these controllers were tested in, I am using Team Delta 18 volt Dewalt powerdrive kits. These motors are capable of drawing up to 155 amps at stall, and while this is somewhat limited by the output capability of my battery packs, the 6s and 5s A123 packs I'm using are certainly capable of pushing these controllers to the limit.

After the impressive performance of these controllers, I intend to continue using them in my robots for the foreseeable future. The small footprint, low weight, and high durability make them a great deal.

The Holmes Hobby TorqueMaster BR-XL speed controllers are available through **HolmesHobbies.com** and retail for $98.88. In addition, they are listed as coming soon at **RobotMarket place.com**. SV

EVENTS

Completed Events for February 2011

BotsIQ/BattleBots 2011 Nationals was presented by BotsIQ in Miami, FL on February 23–27, 2011.

Motorama 2011 was presented by North East Robotics Club, Inc., in Harrisburg, PA on February 18–20, 2011.

SouthEastern Combat Robotics held an event at the Engineering Expo of the University of South Florida in Tampa, FL on February 20th.

NWMHE 2011 was presented by Western Allied Robotics in Monroe, WA on February 12, 2011. SV

EVENT REPORT: 2011 BotsIQ Rocks the Miracle Marketplace

● by Kevin M. Berry

This year's BotsIQ Nationals were a tribute to the creativity and resourcefulness of not only the builders, but the organizers. Forced into a last minute venue change from their roomy traditional location, they smashed — so to speak — the whole chaotic, wonderfully organized mess into a newly built but unoccupied future department store with less than half their usual square footage.

Thanks to the owners of the Miracle Marketplace, the facility was made human safe but robot dangerous in a record three weeks. And the games began! Eighty teams of student builders and five days of robotic competition — much of it carnage based.

BotsIQ is an educational program created by the producers of BattleBots television. Held since 2001, the program utilizes three distinctly different robotic competitions. First, there's the task oriented (table top) competition where operator-controlled robots have to perform specific tasks that often mimic real-life robots such as the Mars Rovers. This year, BotsIQ debuted the use of the popular VEX robotics system for this portion of the event. Middle, high school, and college teams participate.

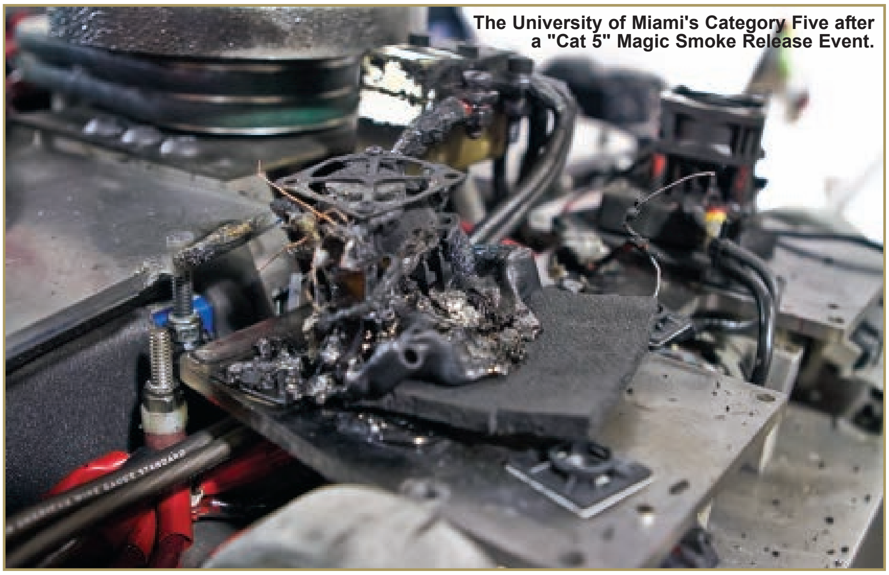
The second and third portions are 15 pound and 120 pound combat robot fights. These are organized by high school, college, and pro class competitors. Like all combat events, these bring in a wide range of experience, from novices to long timers on the national circuit.

SERVO asked a couple of the sport's prominent personalities to describe their best memory of the 2011 event. Greg Munson, co-creator of BattleBots, shares this story.

"The coolest thing that

Bots left to right: University of Miami's Category Five and Polytechnic University of Puerto Rico's Rhino. People left to right: Jaime Ignacio Balzac, Kathiria Cintron, Andrea Suarez, Jorge Torres, and Jesus A. Librada.

University of Texas at Dallas' Blender (left; winner of the College Division) takes on Pro 120 Blue Flame (built by Victor Soto).



SPONSORS

The Florida Engineering Society

The Florida Engineering Foundation

The Florida Institute of Consulting Engineers

Florida Engineering Society Miami Chapter

Army Research Labs from UCF Tri-City Electric

The Talisman Corporation

The City of Miami

StarBot, Inc.

The Symbiosis Foundation

Cargo, Inc.

Scott Masson Construction

METAL MEETS MIRACLES

by Nola Garcia

The Bots IQ 2011 National Robotics Competition in Miami was one more time where the bright technological future leaders of our country came together and revealed some of their best "stuff."

As the over 80 student teams around the country worked together to prepare for this event, other teams of people in Miami also prepared for the event. The teams of students ranged from middle school students from a Catholic grade school in Miami to college students from Colorado, Washington State, Missouri, Minnesota, Texas, and Miami. The eclectic group of people who gathered to celebrate science, technology, and innovation were an inspired group to be sure!

From my perspective as event organizer, it was a miraculous weekend. At the last minute, the venue had to be changed because there was structural damage at the original venue that was not able to be repaired in time for the event. Teams of people all over Miami from the Mayor's office to real estate agents across the county scoured every place they could imagine to find the right combination of specs to fit the criteria needed for an event like this. At my wits' end late one afternoon, I called out to the angels who always seem to come through for me "I need a miracle! There are kids coming from all over the country — they bought and paid for airline tickets, they designed and built their robots! Come ON! I need a miracle here!" Literally two minutes later, I came across The Miracle Marketplace. After meeting with the angelic people who run the Marketplace and their owners, we had a new venue. However, it had its challenges. No electricity, no air conditioning, raw concrete walls ... The list was seemingly endless. But "angels" kept coming out of the woodwork! In the end, the place was transformed into a delightful environment that for four days and nights held some of the coolest matches from VEX task oriented robot competitions to exciting 15 and 120 pound battling robots.

The event was a safe one that left everyone from the competitors to the random viewers off the street exploding with their version of the coolest match where the robots either hit, smashed, or picked up the rings. The Monday after the event, I ran into two people who shared their stories of how the weekend impacted their families. From nine to 90, they were thrilled to be there. A nine year old boy whose father worked for two and a half weeks to get the proper electricity to the venue proudly took his Bots IQ poster to school Monday morning stating "Every Monday I have to listen to all the cool things that my classmates did over the weekend. This time, I am the one who did something way cooler than they did and here's the proof!" A 91 year old lady whose daughter-in-law brought her to see the robots was tickled to see the kids and the amazingly hearty robots they built that were strong enough to take all the hits and flips. She left the competition and proceeded to call her family and friends to tell them that she just spent the most amazing afternoon with battling robots.

We seem to think technology is cold and hard, and has nothing to do with the warmth of humanity. This past weekend was proof that technology is an extension of who we are and what we have created with the gifts and talents we have been given.

happened at this year's BotsIQ had to be the story of Fluffy DeLarge. For the past few years, Will Bales, Alex Mattaway, and the team from Ransom Everglades High School have competed with a robot named Fluffy (among other bots), and every time Fluffy catches on fire. This year was no exception. For 2011, they upped the stakes and made a 120 pound version of Fluffy — Fluffy DeLarge. She did smashingly well in the High School 120 tournament, defeating all bots in her path, only to — thankfully — spark flames at the END of her championship match. Fluffy won the High School 120 competition, but that victory was not enough. Will and the team decided to challenge the Pro 120 champions Witch Doctor. After a flurry of false starts and repairs, the match was on, and in less than one minute, Fluffy DeLarge smacked and overturned Witch Doctor, defeating one of the most destructive and ornery pro robots the BotsIQ competition had ever seen. Way to go Fluffy!"

Marc DeVidts, long time BotsIQ and BattleBots veteran, Event Organizer, and Builder's Database creator, had this to relate:

"The main thing that never ceases to amaze me about the teams at BotsIQ is the incredible sportsmanship. The best example I

can describe is from this event.

I had mentored a team for the last few months as they built a 15 lb robot for this year's competition. The team captain was brand new to robots and his access to tools was very limited. He was really having trouble with his robot at the last minute before the competition. Unfortunately being a volunteer at the event, I had very little time to help my team.

When they showed up to the event, they happened to get pitted right next to another brand new team with another 15 lber from an art school in Puerto Rico. Before I knew it, the team from PR had taken my team's captain under their wing. His robot had not only been fixed, but turned into a winning machine thanks to the advice and guidance of a couple good sportsmen. He went on to win four matches in a row, and he now had a team of 20 Puerto Ricans roaring for him in the bleachers.

You could see that he had been completely transformed by the experience, his face lit up with excitement. The Puerto Ricans had an amazing robot that eventually went on to win the 15 lb division. Never once did they think twice about lending a hand to their competitor along the way, and that is what makes BotsIQ so special and so different from any other competition." SV


Work in the pits is fast and furious.

All photos courtesy of Greg Munson.

EVENT REPORT: Motorama 2011

The Times They are a-Changin' (Bob Dylan)

● by Pete Smith

Each February, the Northeast Robotics Club (**www.nerc.us**) hosts their biggest competition of the year as part of the Motorama motorsports extravaganza and custom car show (**www.motoramaevents.com**) in Harrisburg, PA.

One hundred and twenty robots were entered in a total of seven weight classes from the 250 g Fairyweights to the 30 lb Featherweights, and 95 passed through safety checks and actually competed.

As Motorama is a spectator event and the club needs to put on a show in return for the use of the excellent venue, the two smallest weight classes are fought on the Friday before the main show opens. These weight classes are fun to compete in, but lack the size and action required for a paying audience.

The fights for these weight classes are fought in a small 8 x 8 area (which doubles as a very useful test box for the big bots during the main event.)

The winter weather in Harrisburg is not known to be very kind, but Friday was warm and sunny with a record high in the 60s which made for a pleasant start to the weekend. It was not to last, however. There were only six 250 g Fairyweights, but a record 27 one lb Antweights took part.

While this competition took place, NERC members and volunteers assembled the main 16 x 16 arena. The weight limits for robots are decided by the arenas. The big arena's 1/2" of polycarbonate plastic "glass" is only safe for bots up to the 30 lb Featherweight class. Once this would have been considered safe for even the giant 220 lb heavyweights, but the bots have


FIGURE 1

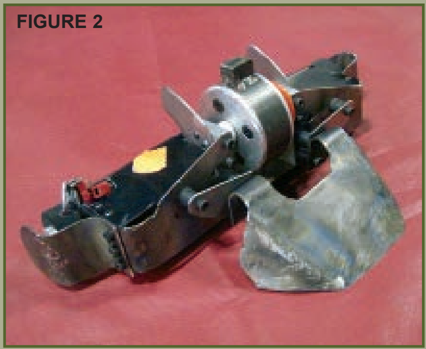


FIGURE 2

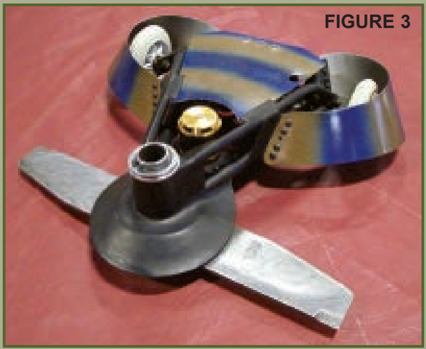


FIGURE 3

evolved rapidly over the years and now even 30 lbers are challenging its limits.

A cold front blew in on Friday night, and Saturday greeted competitors with more typical Motorama weather — 40 mile an hour winds and temperatures around the freezing point. The pits quickly filled up with teams from as far away as Texas, Maine, Michigan, North Carolina, and Canada. There were many local teams as well, with a particularly strong presence from West Mifflin schools.

A new class this year brought a lot of new teams to the event. The Bots IQ Mini Class (15 lb) is part of a school's and college's only competition which has proved to be quite popular. The high cost of traveling to Florida for the competition has led some local teams to look elsewhere for somewhere to compete. The weight of 15 lbs caused some controversy when BotsIQ was created because it is so close to the existing popular 12 lb Hobbyweight class, that it might further reduce the number of bots in any one class. This was not the case this year where both classes had reasonable numbers entered and the arrival of a lot of new school and college teams was very welcome.

This year also saw the return of Team Radicus (**www.teamradicus.com**). They had last competed way back in 2005. Tony Fowlie was famous for two things: the sheer number of bots he entered for an event (I think his record was six), and for the tongue twisting names he gave to his bots. This year, he entered four bots (**Figure 1**): an Ant called Poor Punctuation 1.0, a Beetleweight Didactic Duelist 0.9; a Hobbyweight Cantankerous Cowpoke 3.0; and a Featherweight named Formidable Fustigator1.0. The last featured an arrow shaped chassis and omnidrive. They were all clever and well built bots, but things have changed a lot over the years as he was going to find out.

Most of the teams had their robots ready to go, but as usual, some were getting finished on the morning of the event. Prevaricating seems to be a common tendency in combat robot builders.

All bots have to go through Safety before they are allowed to compete. This consists of a weight test to ensure they do not exceed the limits for the class, a quick check for guards on sharp edges and weapon restraints (important as they prevent the weapon operating unless it's in the arena or test box), and finally a driving test to show that the bot can actually be controlled and to see that the fail safes work if the bot loses the signal from its transmitter. These last two were conducted in the small arena that had hosted the Ants and Fairies on the previous day. (I do not recommend trying to finish your bot and getting through safety all in the couple of hours before the event starts.)

A driver's meeting is then held to ensure everyone knows the rules that apply to the particular competition and a final count is made of which robots and teams are entered.

The fights got underway around 11 am and continued with only a

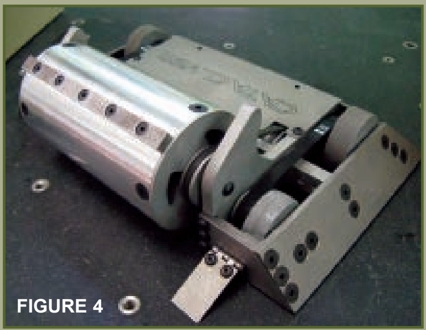


FIGURE 4

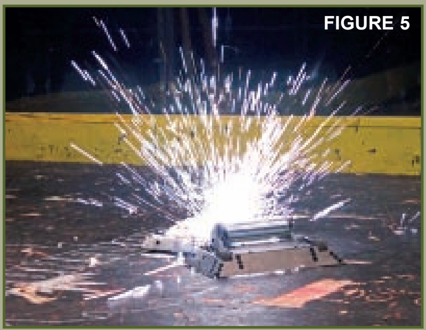


FIGURE 5

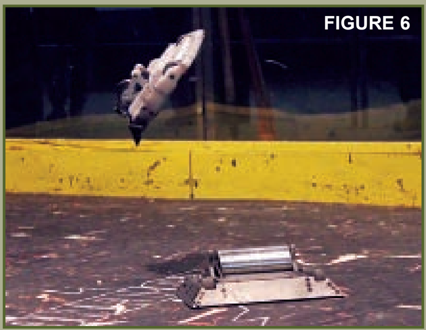


FIGURE 6

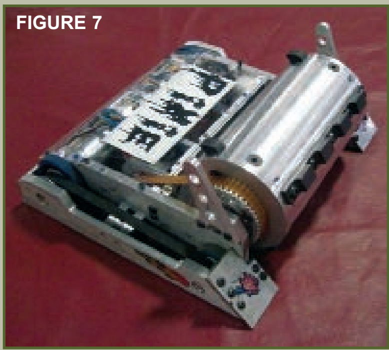


FIGURE 7

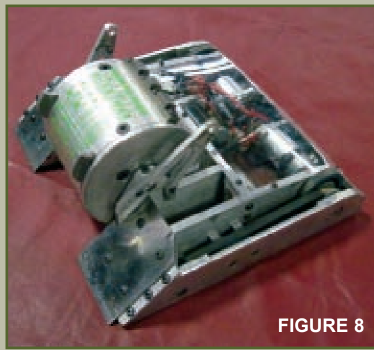


FIGURE 8


FIGURE 9

short break for a late lunch until about 6 pm on Saturday. Then, things restarted the next morning with the last finals held around 4 pm on Sunday.

All weight classes were hard fought. In the Beetleweights, Gene Burbeck's One Fierce Lawn Boy and One Fierce Round House both looked very good until eventually succumbing to mysterious electronic/radio problems. Kitbots', Weta, God of Ugly Things and Devastating Moment both performed well until electronics problems knocked out the latter and the relentless low wedge and sloping sides of Battlebots' toy based D12 beat Weta in a judge's decision. The progress of D12 was only finally halted when it was decisively beaten twice by Mr. Croup (**Figure 2**) who had fought its way through the losers brackets to gain first place.

The winds of change were blowing through the 12 lb Hobbyweights. Last year's champion Scurrie was gone; its creator retired it and built a new vertical spinner, Devour. Gene had entered an innovative bot, One Fierce Dragon Fly (**Figure 3**), that used the rotation of its massive blade to also provide drive (an idea that deserves an article all its own). Two-time champion Surgical Strike was back, but with a basic design now over five years old and the absence of its usual driver, Andrew Smith, it was going to need some luck. The new kid on the block was Cataclysm (**Figure 4**), a very tough looking bot with a four lb drum and sloped titanium sides. This solid design combined with excellent driving was going to prove decisive.

Surgical Strike's luck held for a while with good draws against wedge, Acute Pain, which was "retired" after being torn apart, followed by a barely mobile brick, Obelus, which tapped out after one good hit and finally a very satisfying win against fast wedge, Apollyon. Surgical Strike had lost to Apollyon in the finals at Carolina Combat a couple years ago, but the new hardened steel blade could get a better bite this time and swiftly wrecked the wedge's wheels, so he tapped out.

One Fierce Dragon Fly and Cataclysm met in their very first fight. OFDF got the first big hit in and flipped Cataclysm over which might have been decisive if his next hit had not flipped it right back. Cataclysm then took over and quickly forced Gene to tap out after the aluminum weapon axle failed, stopping the blade and hence all drive. OFDF was out of the competition.

Cataclysm went on to brush aside Devour and Surgical Strike to reach the finals. Devour and Surgical Strike fought in the semi finals but a weapon ESC failure in SS gave Devour an easy win. This resulted in an exciting but fairly one sided final where Cataclysm overpowered (**Figures 5** and **6**) and threw Devour about until he finally tapped out.

The Bots IQ 15 lber rounds were close fought with Pixie (**Figure 7**), Faye-tality, eXecutor (**Figure 8**), and Bar Exam all jostling for supremacy before eXecutor won after an easy victory over a tired Bar Exam in the final.

The Sportsman class was dominated as usual by the superb pneumatic flipper, Upheaval. The power combined with excellent driving to make its first place win over lifter, Gigarange, was no real surprise. There was one new

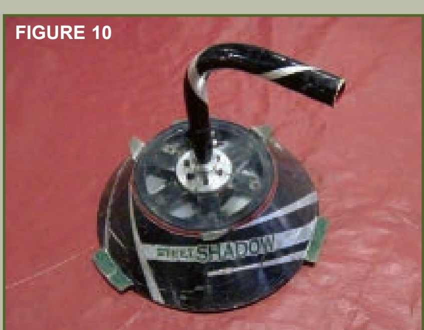


FIGURE 10

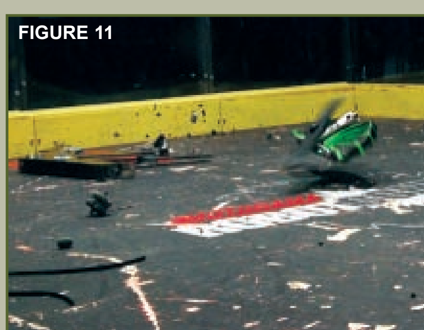


FIGURE 11


FIGURE 12

RESULTS

FAIRIES	1st — Rebound	2nd — lolcat	3rd — Mango Farmer
ANTS	1st — Gilbert	2nd — Jukid	3rd — Night Hawk
BEETLES	1st — Mr. Croup	2nd — D12	3rd — Weta
12 LBS	1st — Cataclysm	2nd — Devour	3rd — Surgical Strike
15 LBS BOTSIQ	1st — eXecutor	2nd — Bar Exam	3rd — Pixie
SPORTSMAN	1st — Upheaval	2nd — Gigarange	3rd — Diabolical Machine
30 LBS	1st — Pinball	2nd — Devour	3rd — Higgins

design that pleased the crowds. It was one half of a planned multi bot, Such and Such (**Figure 9**). The bot uses a scissor jack type mechanism to clamp itself onto its opponent while the other half (a hammer bot, not yet completed) would beat it into submission. I'm not sure the plan would have worked but it was fun seeing how the bot operated.

The biggest and meanest bots that fight at Motorama are the 30 lb Featherweights. The field included the fearsome bar spinners, Sloth and Moros, the vertical spinners, Shaka and General Disarray, a full body spinner, Steel Shadow (**Figure 10**), and the tough steel wedge, Pinball. Tony's innovative Formidible Fustigator quickly proved that construction techniques of six years ago cannot stand up to the heavy blades and brushless power of today. After losing — but surviving — its first fight against Higgins, it had the bad luck of drawing Moros — the biggest blade spinner at the event. It only took two hits to reduce weeks of work to shrapnel (**Figure 11**).

General Disarray lost use of its weapon early on in the event but its replacement with a section of 4x4 timber allowed it to absorb the hits from Moros (**Figure 12**), and then invert and trap it in a corner for the win. The same tactics, however, were not successful against Steel Shadow. The shell spinner went on to beat a near weaponless Shaka in the semi-final to meet the wedge, Pinball, in the final. Pinball had proved impervious to damage and despite the best efforts of the spinner, it went on to clinch first place.

The event managed to finish on time and the prizes supplied by sponsors Team Whyachi, Fingertech Robotics, Kitbots, Holmes Hobbies, and Battlepacks were awarded by no less than five "Miss Motorama" contestants. The presence of so many college teams meant plenty of hands to help with dismantling and loading up the big and small arenas, plus all the other tidying up required in the pits area. The now traditional post Motorama dinner was held at the Texas Roadhouse and a good time was had by all. Despite not winning a single round, Tony expressed the feelings of all when he said he'd "had a blast!" That is one thing at least that has remained a constant at Motorama. SV

*Photos by Pete Smith, Tony Fowlie, Brian Benson (**www.bensonpv.com**), and from the Builder's Database.*


SERVOCITY
A DIVISION OF ROBOTZONE, LLC.
WE HAVE THE PARTS FOR YOUR IDEAS!
Specializing in Servos, Radios, Motors, Pan & Tilt Systems, Batteries, Wiring, Connectors, Gearboxes, and much more!
Aluminum Hitec Servo Sprockets
32 & 48 Pitch Aluminum Hub Gears
GEARS & SPROCKETS
Clamping Tube Hubs
Hub Mount Sprockets
90° Clamping Tube Hub
Clamping Hubs
32 & 48 Pitch Brass Servo Gears
Set Screw Hubs
HUBS & TUBE MOUNTS
Metal & Plastic Chain
Precision Hub Mount Wheels (2", 3", 4" & 5")
WHEELS
Direct Mount Servo Wheels
Foam Wheels (2", 2.5", 3", 3.5" & 4")
Hollow Aluminum Tubing 2" - 12" Length 3/8" & 5/8" Diameter
SHAFTING & TUBING
Precision Shafting 1" - 12" Length 1/8", 3/16", 1/4", 5/16" & 3/8" Diameter
Ball Linkages (mini, standard & super duty)
Gear Motor Gearbox
GEAR MOTORS
Over 20 different Gear Motors available! (3-12V & 12-24V)
Power Gearbox Arms
Threaded Rods
12VDC Robotzone Gearmotors
Threaded Pushrods
LINKAGES
Ranging from 0.5 - 303 RPM
FREE
5 PIECE HORN PACK!
Rod Ends
Enter promotional code 0511FH in the "SCMPC#" field at the time of checkout and receive a FREE HORN PACK with your purchase. Limit one free pack per customer, per order. Offer valid while supplies last or thru May 31, 2011.
To view our entire selection of products, visit us online at
WWW.SERVOCITY.COM
Copyright 1999-2011 Robotzone, LLC. - All Rights Reserved
ServoCity is a registered trademark of Robotzone, LLC.
Phone: (620) 221-0123
E-mail: sales@servocity.com

A Single Axis Robotic Arm With End Effector

Getting a robot arm to lift up and down is pretty easy; just connect it to a servo. The challenge is trying to get it to pick up an object. Taking a clue from the end effector on the Space Shuttle's robotic arm, I came up with a "snare" of an idea. Now, my CheapBot robots carry a practical robotic arm.

The single axis robotic arm described here lifts and lowers targets captured within its end effector. The end effector is a wire snare that wraps around targets, and is extended and retracted by a miniature servo. Because of a momentary pushbutton switch at the end of the arm, a robot carrying this arm can determine when it has successfully captured and released its target. Combined with a little logic, I was able to program a CheapBot to follow a line and use its arm to pick up plastic jacks at one end, and deposit them at the other.

To keep the arm practical, I selected components that were available at local businesses, including hardware stores, hobby shops, and craft stores. I ordered the few remaining items from Jameco. Here are the necessary parts:

- 1 inch diameter hidden pillar*
- 1/8 inch polystyrene plastic tube, 14 inch long (Evergreen 224)
- 1 inch x 8.5 inch sheet of 1/8 inch thick Syntra**
- 1 inch x 4.375 inch sheet of Correplast***
- 6 inch of 3/16 inch square brass tube
- Standard servo (like a Futaba S-148)
- Miniature servo (like a GWS Pico or JR Sport Micro)
- Momentary pushbutton switch (Jameco part number 106112)
- Piano wire, 20 mil diameter, 36 inch long
- Mini EZ connector****
- 4.7K ohm resistor (1/4W)
- Heat shrink
- #24 AWG stranded wire
- 2-56 machine screws
- 2-56 nylocks (use nylocks so the arm can't fall apart from use)
- Connectors to your robot controller (like three-pin headers)

*Hidden pillars are fluted tubes used in wedding cakes; I got mine from George's Hobbies in Lawrence, KS in their cake decorating aisle.
**Syntra is a 1/8 inch thick sheet of foamed PVC (poly-vinyl chloride). Since it's foamed (filled with air bubbles), it's softer and easier to machine than regular PVC (which is a very hard plastic). Syntra is available from online robot stores, local plastic companies, and some sign companies.
***Correplast is corrugated plastic and very popular as a sign material. I was able to purchase some from a local sign company.
****Mini EZ connectors attach piano wire push rods to servo horns in model airplanes. You'll find them in hobby shops that carry RC airplanes.

Putting the Pieces Together

The arm is constructed in five pieces:

1. The arm core.

2. The capture switch.
3. The arm cover.
4. The snare.
5. The arm joint.

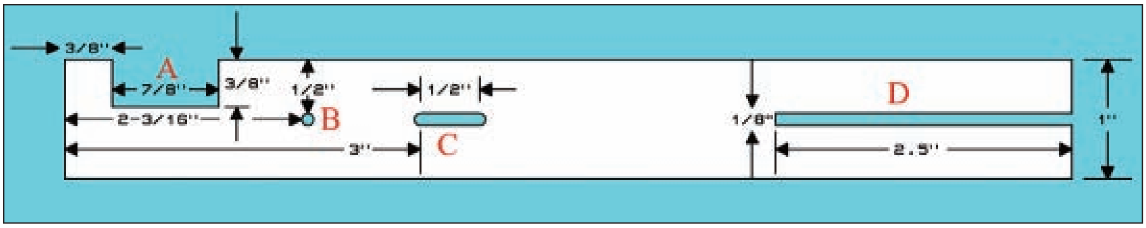


FIGURE 1. These are the dimensions of the robotic arm's Syntra core. Pocket A is the location of the snare servo. The main servo's horn mounts to the Syntra core at hole B. Slot C is the snare wire pass-through and slot D is where the Correplast core slides into the Syntra core.

1. Making the Arm Core

The core of the arm is a cross of Syntra and Correplast. They intermesh at right angles, dividing the interior of the arm into quadrants. So, get a sharp Exacto knife and metal straight edge to cut out the Syntra and Correplast cores as shown in **Figure 1**. Note that the dimensions for pocket A in the Syntra may be different for your miniature servo. The dimensions listed are for the GWS Pico servo.

After cutting out slot C, use a small rat tail file to further shape the slot. The ends of the slot must bevel so the snare wire can pass through without rubbing against the Syntra. **Figure 3** shows how to bevel the ends of slot C.

The great thing about Syntra is that it is easy to cut. The bad thing about Syntra is that at 1/8 inch thick, it's not very stiff. Therefore, we need to attach a brass rail 1/8 inch above the bottom edge of the Syntra to add stiffness. Use 2-56 bolts (1/2 inch long) to hold the six inch long rail to the Syntra core. Note that the rail mounts on the opposite side from the main servo's servo horn. This servo horn (centered on hole B) connects the arm to the main servo (the servo that raises and lowers the arm). Which side you decide to mount the rail isn't important, just as long as the servo horn and rail are on opposite sides of the Syntra core. Before attaching the rail, bolt the center of the servo horn to the Syntra core at hole B. See **Figure 4**.

Mount the rail to the Syntra core by its ends. Then, drill two more holes through the rail, but this time also drill them through the servo horn. Now, remove the servo horn and enlarge hole B in the Syntra core until it is large enough for the screw holding the servo horn to the servo to pass through. Making hole B this large lets the screw bolt the servo horn to the servo without having to pinch the Syntra core. Use two more 2-56 bolts to attach the rail and servo horn to the Syntra core. Then, drill one last hole though the Syntra, passing through the top of the servo horn. Use nylocks when you bolt the servo horn and rail to the Syntra core. Now, slide the Correplast and Syntra cores together as far as it will go. The end of the arm core will look like the **Figure 5**.

Next, cut the raceways. These are 1/8 inch plastic tubes; one five inches long and the other six inches long. After cutting the raceways, mark each one 5/8 inch from one end. The raceways sit inside the open channels of the Correplast and extend 5/8 inch beyond the end of the arm core. The five inch long raceway is on the servo horn side of the arm and the six inch long raceway is on the rail side of the arm. See **Photo 1**.

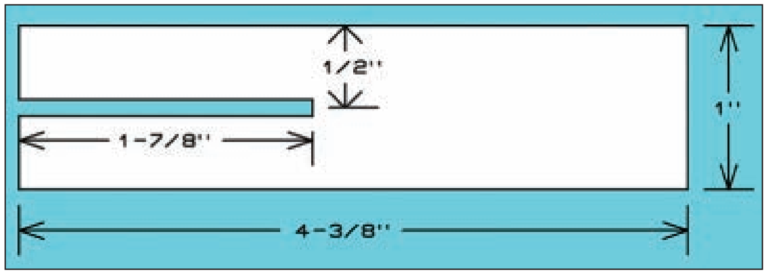


FIGURE 2. The slot cut through the center of the Correplast is 1/8 inch wide, the same as the thickness of the Syntra. The slots permit the Syntra and Correplast to slide into each other to form the core of the arm. Cut the Correplast with five channels running down its length. The center three are full channels and the outer two channels are open. The raceways for the snare wire are glued to these two outside channels.

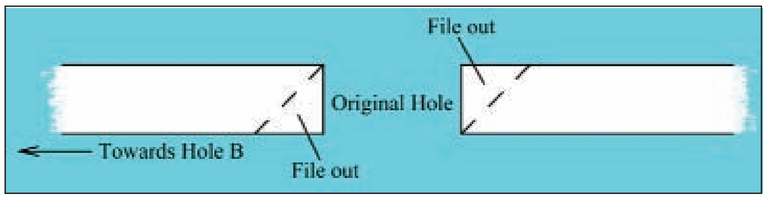


FIGURE 3. Slot C was cut and drilled perpendicular to the Syntra core, but must be beveled slightly in order for the snare wire to pass through it without interference.

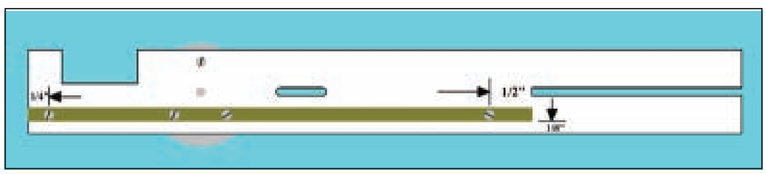


FIGURE 4. Drill the holes 1/4 inch and 1/2 inch from the ends of the rail and then bolt the rail to the Syntra core. You'll drill the remaining holes next.

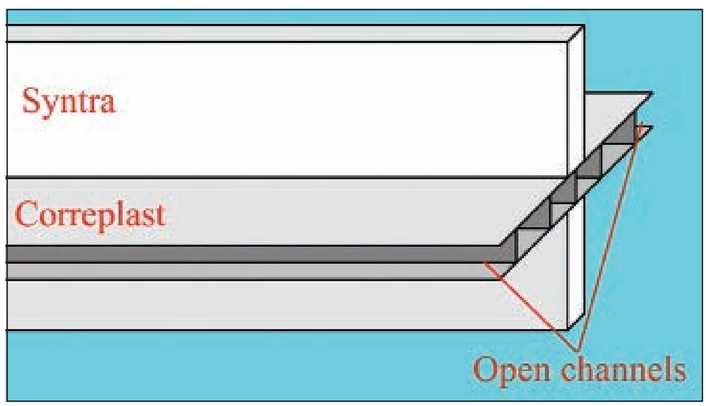


FIGURE 5. The cross of the Syntra and Correplast cores creates the quadrants inside the arm that the hidden pillar will cover. Also, notice that the outermost channels of the Correplast core are opened.

PHOTO 1. The raceways extend 5/8 inch beyond the end of the Syntra and Correplast cores. Do not apply hot glue to the Syntra and Correplast within 1/2 inch of the end of the arm (I regretted doing so for this arm).

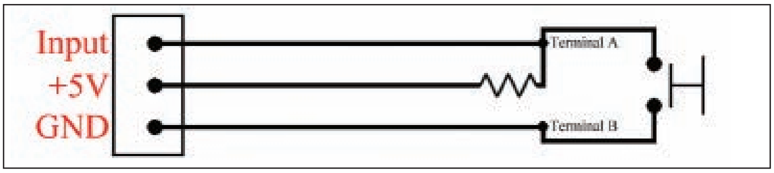


FIGURE 6. Just twist bare wires and solder them to the first terminal of the switch. The soldering will remain inside the robotic arm where it is safe from and invisible to the world.

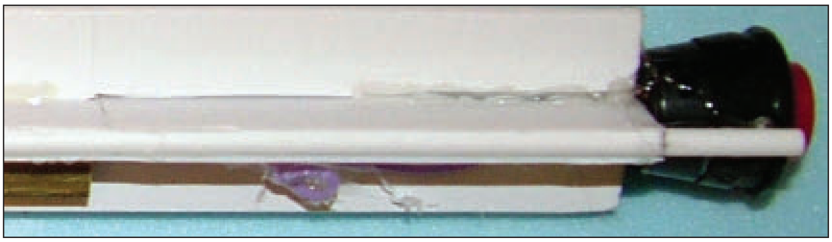


PHOTO 2. After gluing this switch to the end of the robotic arm, I routed the ground wire through two holes on the core to bring it next to the +5V and input wires.

2. Building the Capture Switch

The capture switch indicates to the robot controller when the snare has lassoed the target. That permits a robot controller to "know" it successfully picked up the target before advancing to the next step of its program. The switch is a momentary pushbutton switch mounted to the end of the robotic arm; three wires connect to it. An opened switch permits a resistor to pull up the robot controller's input to five volts. A closed switch shorts the robot controller's input and resistor, taking the robot controller's input to ground.

The first step is to cut the three pieces of wire that will connect the arm to your robot controller. The wires connect to +5 volts, ground, and an input to the robot controller; therefore, you may want to color code the wires. Next, solder one lead of the 4,700K ohm resistor and one end of the input wire to a single terminal of the switch (labeled Terminal A in **Figure 6**.). Solder the ground wire to the other terminal of the switch (labeled Terminal B in the diagram). Then, hot glue the switch to the end of the arm and between the raceways. Lay the pull-up resistor (and its +5V wire) and input wire inside one quadrant of the arm core. The ground wire runs down the opposite quadrant. It's up to you, but you might want to drill two small holes in the arm's core so the ground wire can be routed to the same arm quadrant as the +5V wire, pull-up resistor, and input wire. Check out **Photo 2** and **Figure 7**.

3. The Robotic Arm's Cover

Slide the hidden pillar over the arm core (no frosting on the hidden pillar, please). Slide it until its end is flush with the black ring of the switch. This permits the red button to protrude slightly beyond the end of the arm sheath. To keep the sheath in place, squirt a little hot glue between the hidden pillar and the capture switch. However, do not let hot glue get into the raceways.

4. Making the Snare

Bolt the miniature servo (the snare servo) to Pocket A of the Syntra core with 2-56 bolts and nylocks. Locate a servo horn for this servo that has just two arms — you don't need the disk-shaped servo horn. Then, mount the mini EZ connector to the end of one arm of the servo horn.

There are three parts in the mini EZ connector: a black plastic cap, set screw, and metal barrel. There's a thin metal pin on the barrel, opposite the set screw. The pin fits through the holes in the servo horn, so you don't need to drill holes. Slip the metal pin through a pre-drilled hole from the underside of the servo horn and squeeze the black plastic cap onto the barrel's pin. Do not squeeze the black cap too tightly; you want the mini EZ connector to spin freely on the servo horn. Remember, don't squeeze the cap too tightly, and mount the barrel and set screw mounts to the underside of the servo horn as shown in **Figure 8**.

Wear safety glasses while building the snare since the piano wire can get loose and whip around. Cut 22 inches of

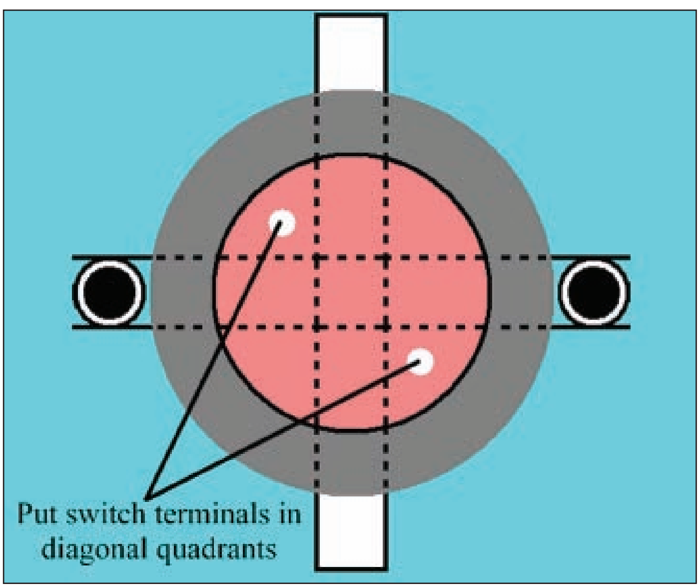
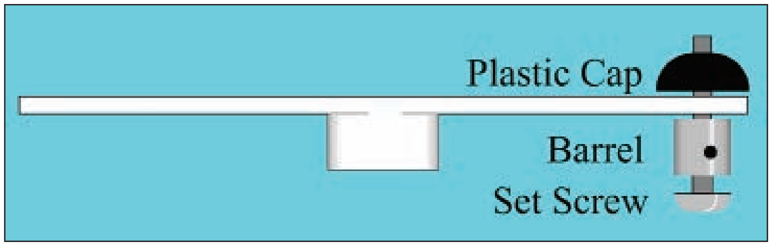


FIGURE 7. A view from the end of the robotic arm with the capture switch. Notice the switch's terminals are located in opposite quadrants of the arm core. Also, notice the switch fits between the raceways.

PHOTO 3. This image comes from later in the construction process (your snare is not in place yet). However, it shows how the sheath is glued to the end of the arm.

FIGURE 8. Use small pliers to squeeze the plastic cap onto the barrel's metal pin.



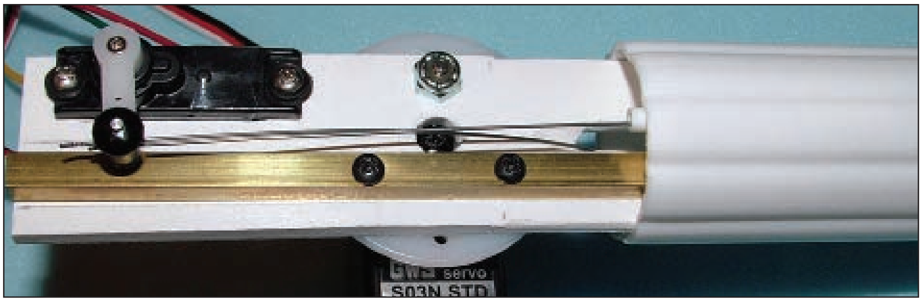


PHOTO 4. Both ends of the snare wire are trapped within the barrel of the mini EZ connector. Now when the snare servo rotates, it extends and retracts the snare at the far end of the robotic arm.

piano wire and bend the snare wire in half without putting a fold or crease into the wire. Then, slide the ends of the wire into the raceways (the openings next to the capture switch) until the ends of the wires protrude from the other end of the raceways (near the snare servo).

The snare wire from the five inch long raceway must be diverted through the arm core before it can reach the snare servo. Use needlenose pliers or tweezers to route the wire through Slot C, being careful not to bend or crease the wire.

Remove the servo horn from the snare servo and loosen the set screw in the mini EZ connector. Slide both ends of the snare wire through the small opening in the barrel. For now, tighten the set screw, but know that the snare still needs adjustment.

5. The Arm Joint

Center the main servo in its rotation. Then, remove the screw from the main servo and push the servo onto the servo horn that you've already bolted to the Syntra core. Drop the screw through Hole B of the Syntra core and tighten the servo horn and robotic arm to the main servo. The arm is now complete; it just needs to be mounted to the robot and have the snare adjusted.

Connecting the Robotic Arm to a Robot

How you attach your robotic arm to the robot — electrically and mechanically — depends on your robot and robotic controller. So, decide where to mount your robotic arm, and then look at the cable lengths and terminations (for both servos and the capture switch). I have had success attaching the robotic arm to a robot base using a short aluminum rail as shown in **Photo 5**.

If your robot controller is similar to those in a CheapBot robot, you will need to terminate the ends of the servo wires and capture switch with three-pin headers. However, if your robot controller has connectors specifically for servos, then you may only need to extend the length of the servo cables to reach the robot controller.

Adjusting the Snare

The snare servo doesn't rotate much more than in a 90 degree arc. At one extreme, the servo horn (and the mini EZ connector) rotates towards the far end of the arm, extending the snare. At the other extreme, the servo horn rotates back towards the nearer end of the arm to retract the snare. Now, we just need to adjust where the snare wire fits into the mini EZ connector barrel and cut off the excess piano wire.

First, make sure the snare servo is centered in its rotation. Then, remove the servo screw for the snare servo and take off the servo horn. Loosen the mini EZ connector's set screw slightly and then put the servo horn back onto the snare servo with the servo horn in the mid point of its swing (straight down from the servo and towards the bottom of the arm). Pull or push both ends of the piano wire through the mini EZ connector until the snare is about large enough to slide your little finger through the loop.

Now, you need to

PHOTO 5. A short length of aluminum rail seems to make a very solid base for this servo. The bolts attaching the rail to the robot base are 4-40s and the bolts attaching the servo to the rail are 2-56s.

remove the servo horn from the servo without letting the snare wire slip through the barrel. So, grab the snare wire and mini EZ connector tightly before lifting the servo horn off the snare servo. Tighten the set screw and put the servo horn back on the snare servo. Before screwing the servo horn back onto its servo, write test code to exercise the snare servo. Extend and retract the snare to verify it opens and closes well. When the snare is closed, it cannot press on the switch. The retracted servo needs to leave a little space between itself and the switch button so that only when a target is captured can the capture switch be pressed. Make additional adjustments to the snare wire and set screw until the snare opens and closes properly. Once you're happy with the snare's size, attach the servo horn to the servo with its screw. Hold onto the ends of the piano wire as you cut off the excess wire. Hold the wires to make sure they don't go flying some place dangerous, like you eyes.

Using the Robotic Arm

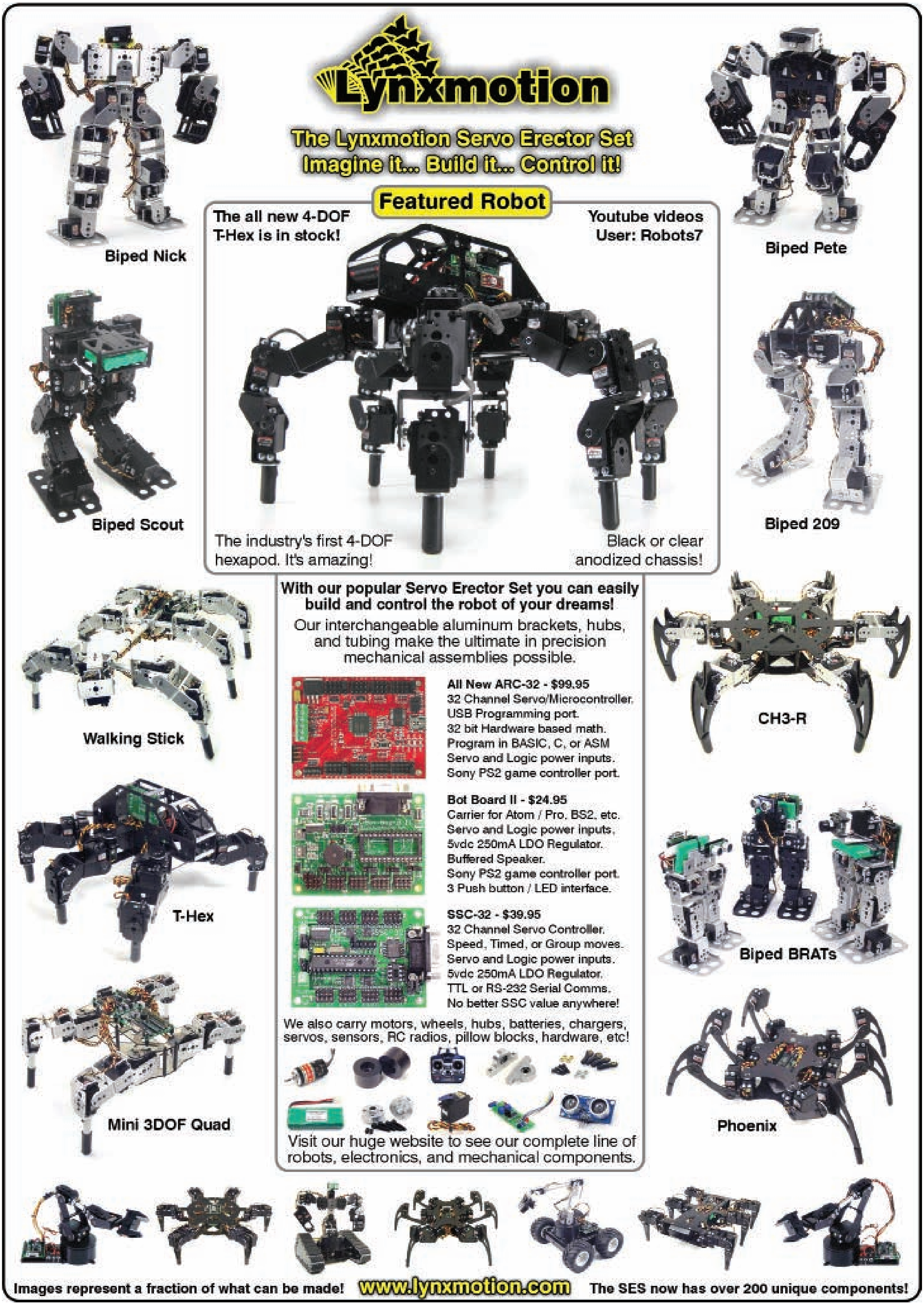
Good targets for the robotic arm and its snare have risers on them, like dowels. You might try lightweight cubes and glue dowels to one face. Right now, I'm playing with large plastic jacks. I like their design since no matter which way they fall, they always have a riser standing straight up.

Now, program your robot for a challenge. Be sure to raise the robotic arm before moving your robot. That way, the arm isn't driven into the wall. Once at its destination, stop the robot, extend the snare, and lower the robotic arm. If the snare has lassoed the target, when it retracts it will pull the target against the capture switch. After retracting the snare, the robot controller checks to see if input from the capture switch is low (ground or logical zero). If it is, then the target has been captured and the arm is ready to lift the target up, and the robot is ready to drive to its next destination.

If the input from the capture switch is still high (+5 volts or logical 1), the snare has missed the target. In this case, the robot should open the snare and raise the arm a small amount. Rock the robot back and forth, and repeatedly lower the arm and retract the snare. At some point, if the robot continues to fail, it should back away and reapproach the destination. Once the robot has driven the target to the second destination, the snare only needs to extend to release the target. The release of the target is confirmed by checking the input from the capture switch to see that it is high (+5 volts or logical 1).

Another option is to mount the robotic arm on its side. In this position, the arm swings left and right to lasso dowels on the side of its targets. The robot would then have to lift slightly to raise the target before moving to a new destination.

You can see a short video of an earlier version of this robotic arm picking up jacks on my website. I am also selling a complete robot arm kit for roboticists who don't want to go to the trouble of locating all the parts for the arm. Find out more on my website at **NearSys.com/catalog**. SV


Lynxmotion
The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!
Featured Robot
The all new 4-DOF T-Hex is in stock!
Youtube videos User: Robots7
The industry's first 4-DOF hexapod. It's amazing!
Black or clear anodized chassis!
Biped Nick
Biped Pete
Biped Scout
Biped 209
Walking Stick
CH3-R
T-Hex
Biped BRATs
Mini 3DOF Quad
Phoenix
With our popular Servo Erector Set you can easily build and control the robot of your dreams!
Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.
All New ARC-32 - $99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.
Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.
SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!
Visit our huge website to see our complete line of robots, electronics, and mechanical components.
Images represent a fraction of what can be made!
www.lynxmotion.com
The SES now has over 200 unique components!

A UART/SPI Monitor For Your Micro

by Mark Mitchell

Probably the most important thing when trying to debug robot problems or any microcontroller based systems is VISIBILITY! If you can't see it you, can't fix it.

The objective of this project was to create a tool to increase visibility into the operation of a microcontroller — and ultimately a robot — by getting access through either the serial port or the SPI port, both of which are pretty ubiquitous fixtures on most micros.

www.servomagazine.com/index.php?/magazine/article/may2011_Mitchell

I am sure all microcontroller programmers have used some sort of print statement out a serial port to see what is going on at one time or another. Before in-circuit emulators were available, it was pretty much the main way of debugging. The SPI port is another place that one would like to know what is happening, especially when things are not working properly (which always seems to be at the most inconvenient time).

To gain better visibility into both of these ports (and because I was tired of digging around for that serial to RS-232 board for the thousandth time), I decided to build a tool that could be used on a standard UART to provide the tried and true debug printing to a PC. It would also display the SPI data out (DO) line activity and could inject an SPI data stream to provide a known good data stream for SPI data in (DI) exercising. The SPI monitor also allows me to avoid having to hook up and decode the SPI signals on an oscilloscope — a tool which may not always be available.

Since this project was pretty small, I decided to build it using the BoardworX universal prototyping system which

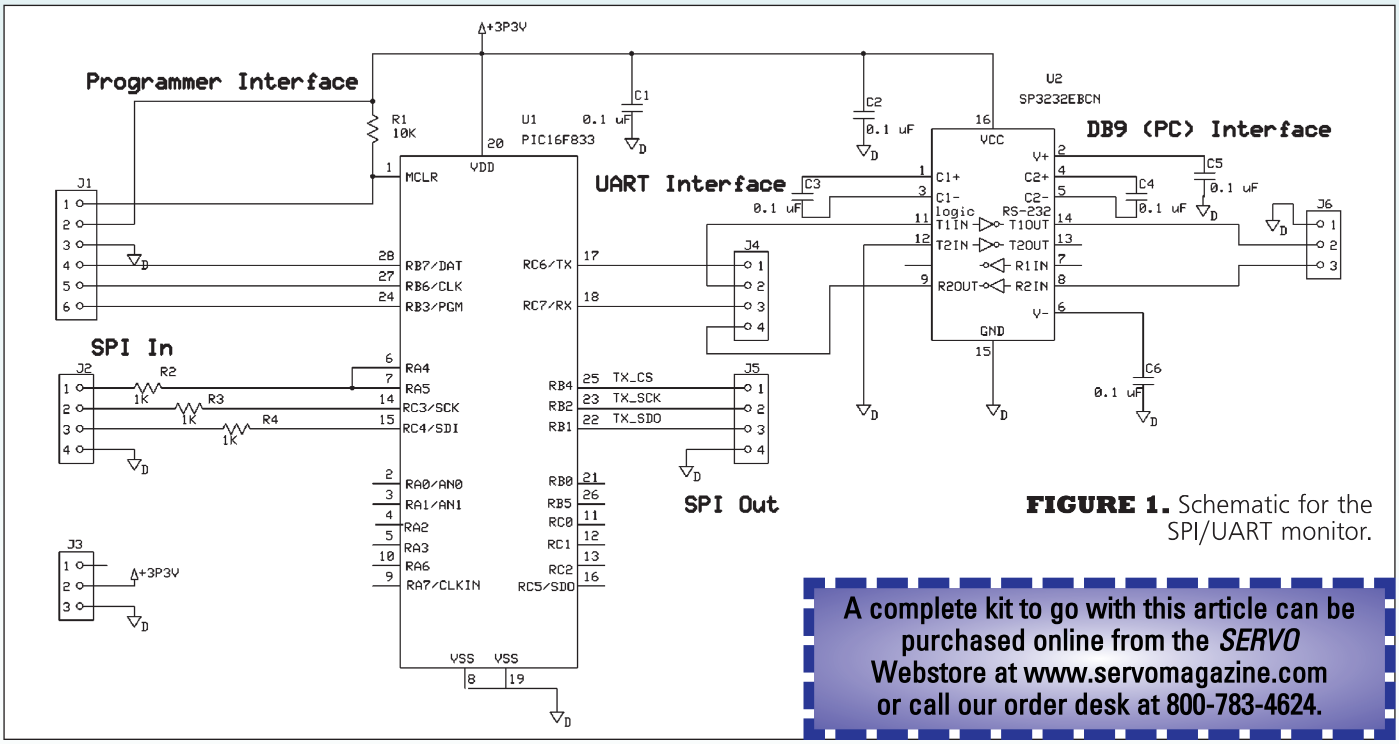


FIGURE 1. Schematic for the SPI/UART monitor.

A complete kit to go with this article can be purchased online from the *SERVO* Webstore at www.servomagazine.com or call our order desk at 800-783-4624.

allowed me to whip up the circuit quickly without having to go through board layout and ordering. The system is designed to accept any SMT part (except BGA) on a single board — regardless of pitch or device size — so I was able to scrounge the parts out of my parts bin and wire up the circuit right away.

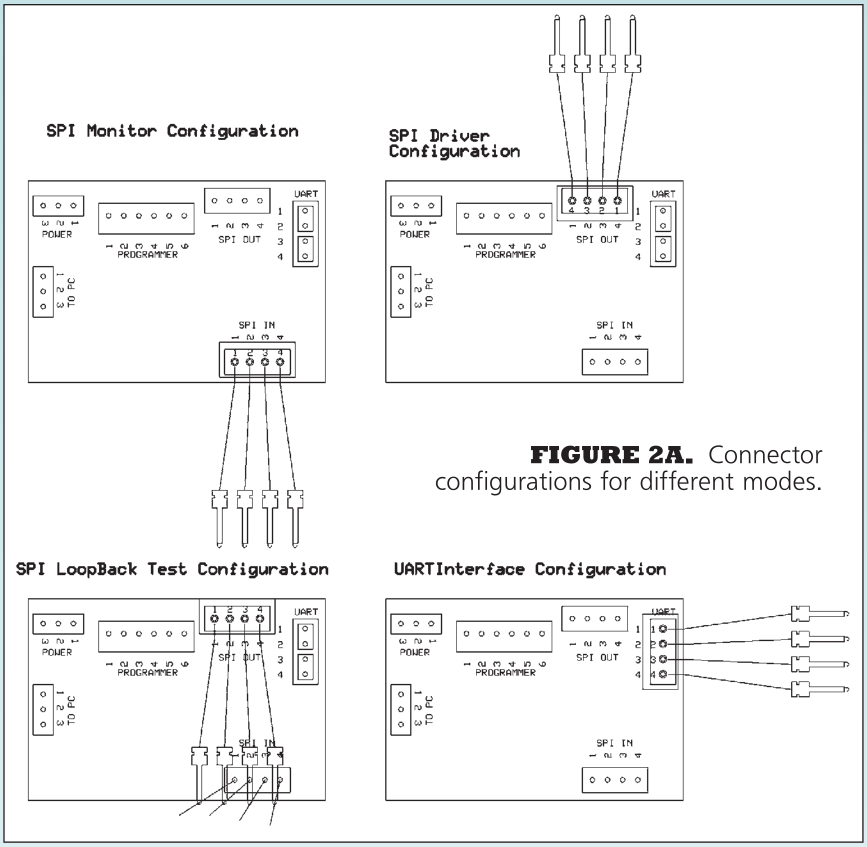


FIGURE 2A. Connector configurations for different modes.

Goals

One goal of this project was to have a simple board that interfaced to a PC via RS-232 (an ancient, but simple and useful protocol) and have it connect to the serial stream from either the UART port or the SPI data stream of a micro under test. In this way, I could see what my bot was trying to do and figure out if it was working properly. On the PC end, having a very simple interface (translation: free) implied using HyperTerminal or some other terminal program. These programs allow capture of the serial stream and provide the ability to log it, as well. The connection was to be made by a simple "hooking in" to the signals of interest on the target system.

Since our old friend the serial port has disappeared from the modern computer — leaving us with its super fast but much more complex successor, the USB port — I used a serial to USB adapter cable to handle the conversion. These adapters allow the serial stream to appear as a simple COM port in the terminal program. One bonus feature is the ability to locate which serial port you are talking to with the terminal program on the PC by connecting the UART/ SPI monitor tool to the USB port (via the RS-232 adapter) of the computer and observing which port responds.

Hardware

The schematic appears in **Figure 1**. Let's look at the SPI path first. The SPI signals originate at the target and come in connector J2 (through the probe connector). The PIC accepts the SCK, DI, and SS (slave select) signals, then converts these to a displayable form and outputs the result to the UART of the PIC (and ultimately to the PC via the RS-232 translation chip). When operating as an SPI source, the probe connector is placed on J5 which presents the SPI signals as outputs.

I decided to have the target circuit provide power to the monitor board because of the possibility of running on 3.3V or 5V targets. This approach allows either without any confusion about what is powering what, and avoids the potential of "letting the smoke out" of target devices by inadvertently having the wrong voltage in the wrong place at the wrong time (which never happens to me, of course...). Power is applied through J3. The board is designed to serve several functions as discussed, and **Figure 2** shows how this is accomplished. Looking for the easiest way to do things, I decided to make a single probe connector and attach it to the different function headers (SPI-IN, SPI-OUT, UART) so that the signals could easily get on and off the board. When using an SPI operation function, the UART header has jumpers to pass the signals from the PIC to the PC. When using the UART monitor mode, the probe connector goes right on the UART header.

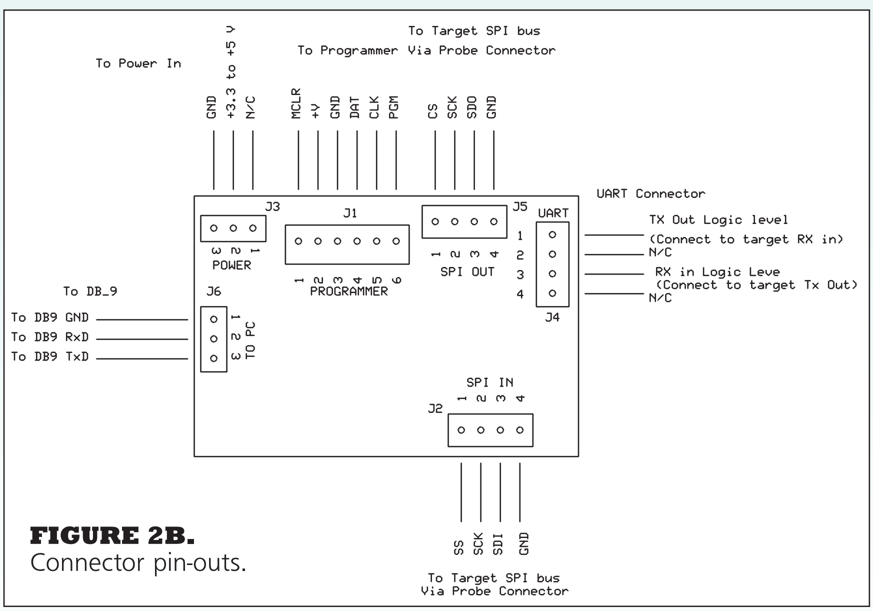


FIGURE 2B. Connector pin-outs.

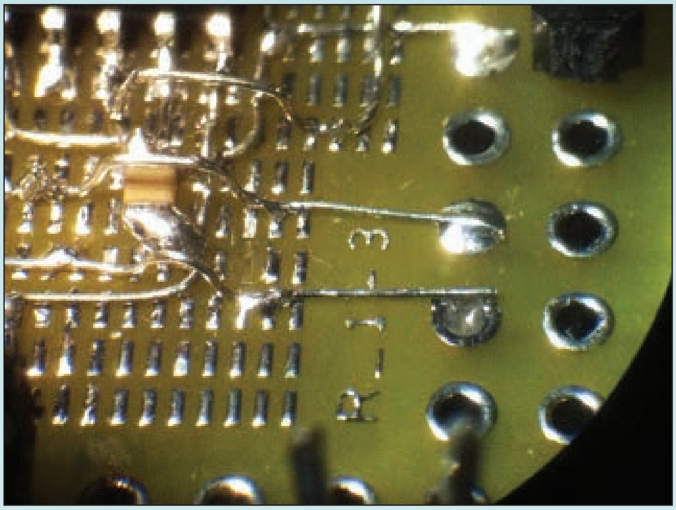


FIGURE 3. Example of wire hopping.

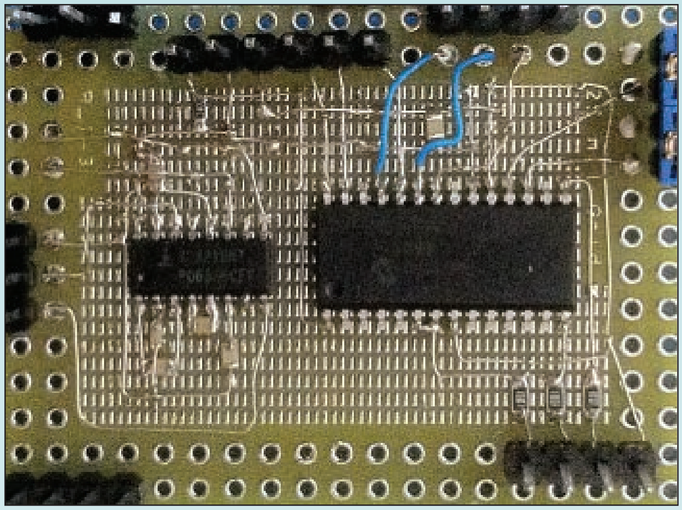


FIGURE 5. The completed board.

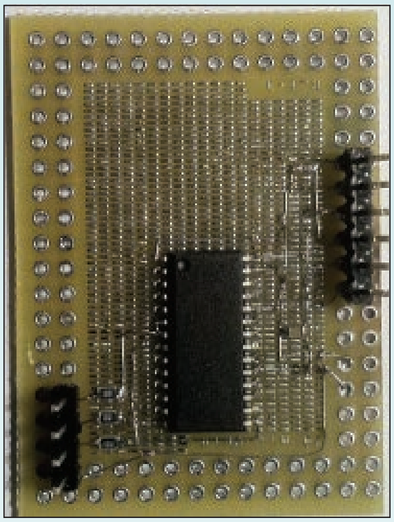


FIGURE 4. Partial completion of the board. This photo shows the routing of the wires and mounting of the parts fairly well.

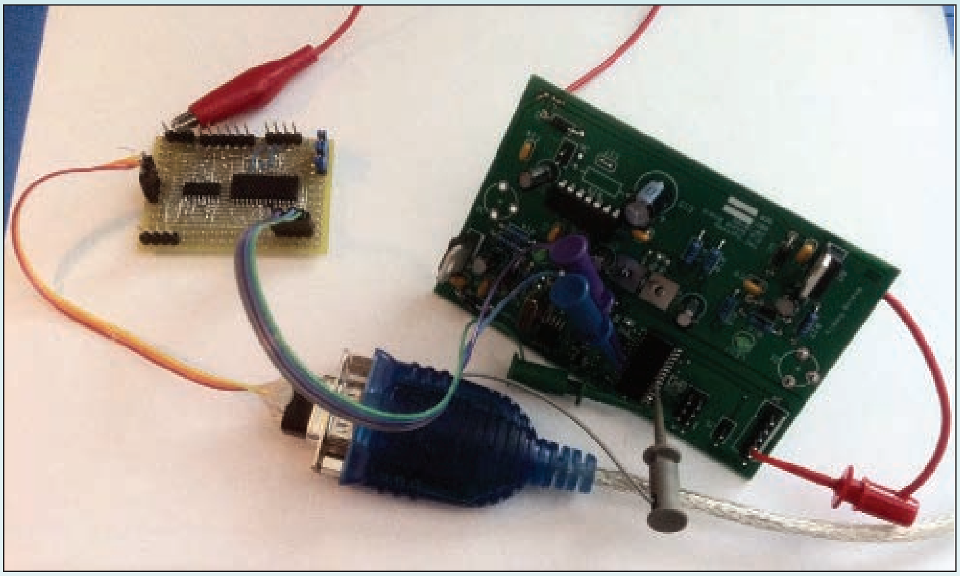


FIGURE 6. Note how the monitor is connected to a target board.

I chose a BoardworX combo board to build the circuit on because it has rows of 0.100" holes to mount connectors in, and I had some 0.100" headers and connectors available. However, surface-mount headers could have just as easily been used. I used 30 AWG bare hook-up wire and 30 AWG insulated wire, as well.

Board Construction

The board construction is quite straightforward. I started by cutting the BoardworX board in half so that the through holes surrounded the prototyping area I would be working with.

I then mounted the microcontroller on the 25 mil pitch side of the board since it fit the pitch perfectly, and tacked down a couple of pins. I then applied solder to the appropriate pins I would be using. I used solder with water soluble flux since I find it is easy to clean up afterwards and non-toxic. I then mounted the passive components and hand-wired each connection.

I used a technique for wiring where bare wire was "hopped over" other bare conductors, which works well with this type of board. One added advantage to this type of wiring (and board) is that most connections are on one side of the board so checking connections is quite fast and easy. One additional note is that because the board is universal, the placement of parts is not constrained so it was easy to find locations for the parts that made the wiring simpler, as well.

I split this process into two steps at this stage and wired in a little RS-232 translator board that I had hanging around since I knew it worked, and this way could get the SPI processor part of the project working more easily. I developed the code at this stage, then added the translator circuit on-board which eliminated confusion about what was working and what was not. **Figure 4** shows the board at the partial completion stage.

I also implemented the SPI transmitter port so that the unit could drive SPI signals out, and so by looping back the SPI out to the SPI monitor it could self test (more on that in the testing section). **Figure 5** shows the completed board, and **Figure 6** shows how it is connected to a target board.

SDI in a Nutshell

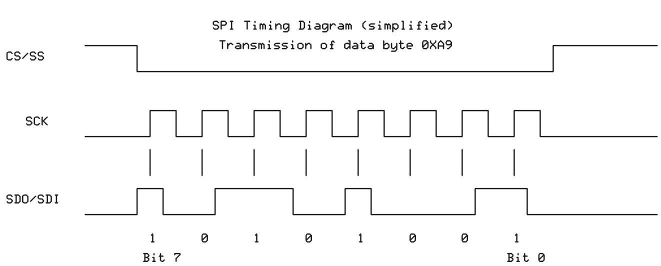
SDI is a protocol for communicating between intelligent digital devices for the purpose of passing information and control.

SDI is what is known as a synchronous serial protocol which simply means that it has a clock signal sent along with the data that is being transmitted. There are four signals typically: SDI (Serial Data in), SDO (Serial Data Out), SCK (Serial Clock), and CS (Chip Select). **Figure A** shows a typical structure and timing diagram of the relationship of these signals.

So, when one wants to transmit a byte from the master device to the slave device, one loads the SPI output register with the byte of data and this initiates the transfer. Each bit is presented to the output pin (SDO) by the master one after another with each bit accompanied by a transition of the clock signal (SCK) also driven by the master; in this case, the transition is from low to high. Each clock transition from low to high tells the slave to latch in the current bit that appears on its SDI line which is connected to the SDO line of the master. In this way, the entire byte is transferred from the master to the slave.

The CS select signal is used to select which slave device the master is talking to since one master can have multiple slaves by connecting the SDO line of the master to the SDI lines of all the slaves together. The selection is accomplished by the master "selecting" the appropriate device (for example, a sensor device) by driving its CS line low and leaving any other device in the system high. Now the data appears on all of the SDI lines of all the slave devices, but only the one that is low (selected) receives the data.

This system provides a very simple and generally fast way for different devices to talk. It is pretty straightforward in software and easily viewed on a scope or analyzer.



CS/SS Goes Low to select the device
Data is presented on SDO connnected to SDI
SCK transitions High to latch in Data bit at transition time
Process repeats 8 times
CS/SS goes high to unselect this device

FIGURE A.

Figure 7. Code listing.

```
  Code Listing   -
//  ------------------------------------------------------------
// MAIN
//  ------------------------------------------------------------
// Function   - This is the program Main function which
// controls the overall operation of the system
//  ------------------------------------------------------------



void main()
{
       // definition and init of increment variable
       unsigned int8 OutByte = 0;
       // Call Init function
       Init();

       // Output initial header
       printf("SPI/UART Monitor Ver 1.0 \r\n \r\n");
       printf("Waiting for data...\r\n");

       //     Setup hardware SPI function
       setup_spi(SPI_SLAVE );

       //     loop forever
       while(FOREVER)
       {
              // Output Byte
              spi_xfer(SPI_OUT,OutByte,8);
              // inc output byte to next value
              OutByte++;

              // check if a byte has arrived at SPI input
              if( spi_data_is_in() )
              {
                     // get the byte
                     Data = spi_read();

                     NewLineCount++;
                     if(NewLineCount >= 16)
                     {
                            // output the byte out the uart plus
                            // return to align to 16 symbols
                            // wide printout
                            printf("|%x\r\n", Data);
                            NewLineCount = 0;
                     }
                     else
                     {
                            // output the byte out the uart
                            printf("|%x", Data);
                     }
              }
       }


} // end program
```

Software

The firmware for this project is quite straightforward, thanks to the good people at CCS who have written a compiler that is economical and easy to use. I first upgraded my Microchip MPLAB environment version to the latest one, then created a new project and started writing the code.

The firmware essentially consists of a loop that runs continuously and outputs an incrementing number on the SPI output port, then polls the SPI port waiting for activity. First, the appropriate ports are set up and the SPI system is initialized. This design uses the hardware SPI port for monitoring because of the built-in slave select mechanism that is provided, and also because it is the fastest (most

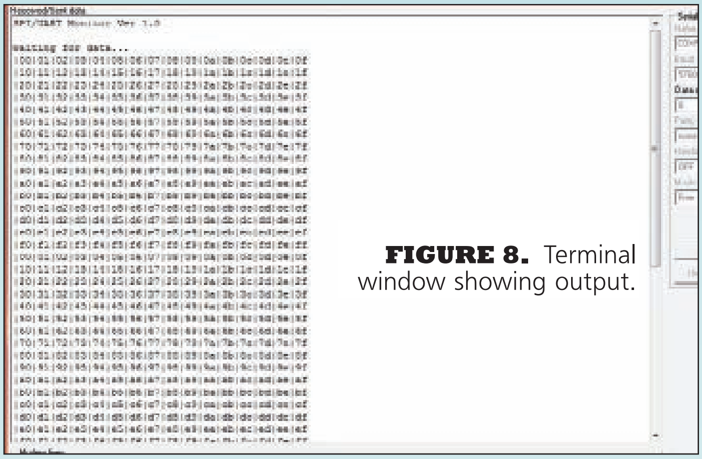


FIGURE 8. Terminal window showing output.

Parts List

ITEM	DESCRIPTION	PART #	SOURCE
Board	SMT-300 Combo Board	N/A	**Boardworxsystem.com**
	SMT-300 Combo Board	2117296	**Jameco.com**
U1	PIC16F883-I/SO	PIC16F883-I/SO-ND	**Digi-Key.com**
U2	SP3232EBCN	1016-1095-5-ND	Digi-Key
	MAX3232ESE+	2053392	Jameco
R1	10K Resistor	311-10.0KHRCT-ND	Digi-Key
		1877832	Jameco
R2, R3, R4	1K Resistor	311-1.00KCRCT-ND	Digi-Key
		1885445	Jameco
C1-C6	0.1 µF 50V Cap	478-3351-1-ND	Digi-Key
		2079576	Jameco
J1-J6	Header Breakable Strip	929400E-01-36-ND	Digi-Key
		68339	Jameco
J7	Connector, four-pin	S7037-ND	Digi-Key
	Connector, three-pin	S7036-ND	Digi-Key
J8	Connector, six-pin	S7004-ND	Digi-Key
H1-H4	Micro Probe Kit	923848-ND	Digi-Key
or	Micro Probe Red	131C307	**Mouser.com**
	Micro Probe Black	131C308	Mouser
	Micro Probe Blue	131C309	Mouser
	Micro Probe Green	131C310	Mouser
	Micro Probe Yellow	131C311	Mouser

Sources

CCS Compiler
www.ccsinfo.com

BoardworX System
www.boardworxsystem.com

Microchip
www.microchip.com

Project Details
markmitchell@mx2 systems.com

responsive) approach. The built-in "soft" SPI functions are used for the SPI transmit output part, as seen with the SPI_XFER function in the source.

An incrementing count (OutByte) is presented to the SPI out port. Then, activity on the SPI in port is polled and when detected (a byte has been received), it is read from the SPI hardware and stored in a temporary variable. The value is then output via the venerable printf statement in a hex format. This goes on until power down or reset. **Figure 7** shows the code listing. The main limitation of the system is speed. The particular processor used has a built-in clock which I used for simplicity and convenience. It is limited to 8 MHz, so the fastest SPI clock signal that can be used on the target is 2 MHz. Additionally, the UART max output speed is 57600 baud which limits the throughput of valid bytes readable on the SPI bus.

The output that appears in the terminal window is shown in **Figure 8**. Note the incrementing count. This capture was done in loopback mode. The lines are 16 bytes across to allow clear display of the data. Each entry in the table represents one eight-bit byte.

Testing

Testing the board was fairly straightforward. Once the unit was constructed and the firmware complete, I connected the SPI output pins to the SPI monitor pins (SPI Loopback Mode in **Figure 2**), applied power and the serial port adapter to my trusty laptop, and watched the traffic flow. The SPI transmitter sends out an incrementing count on each successive byte so by capturing the count and observing it on the PC, I confirmed that the unit and the UART were working correctly. The easiest way for anyone building this project is to download the code from the website and run it with the unit connected in loopback. You can connect it to an external UART or SPI port of course as well, to verify the operation.

Future Expansion

In Part 2, we will upgrade the firmware to include a FIFO (First in First Out) buffer to improve the speed of the monitor. In addition, setup of the different SPI data configurations will be implemented and we will put it in a box to round out the capabilities of the unit. **SV**


robotshop.com/eu
CANADA
USA
EUROPE
RB-1
Now Serving Europe
with Optimized Logistics
Currency in EURO
3 day shipping to 70% of the territory
Service in French and English
Competitive shipping rates
Huge product selection
One Global Door for Manufacturers
VISIT:
Shipping Worldwide
www.RobotShop.com
Robotics at your service! ™

Making Robots With The *Arduino*

By Gordon McComb

Part 7 – Putting It All Together

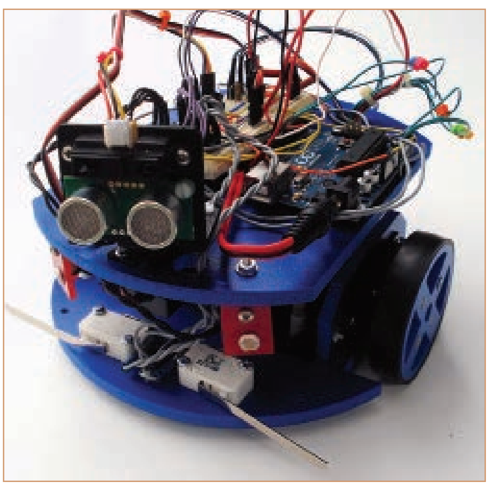
It was the 1970s when I first started playing with robotics. Back then, it was bell bottom pants, disco, and robots that used hard-wired circuits. If you wanted to change the functionality or behavior of your bot, you had to pull out the trusty soldering iron.

Microcontrollers changed all that. Instead of altering hardware, you just revise a line or two of programming code. It's easier, faster, and you're not as likely to burn your fingers. You definitely don't have to make as many trips to RadioShack — in your bell bottom pants or otherwise.

What's even better than using microcontrollers with your robot? Using cheap microcontrollers, of course! The open source Arduino is a leader in low cost microcontrollers. For about $30, you get a complete development board. Just plug it into your computer via USB, fire up the free software, and start coding.

For the last six months, I've been writing about the ArdBot — an expandable and affordable robot test platform that uses an Arduino Uno as its brain. Previous articles have discussed core concepts and have introduced important motor and sensor interface techniques. In this seventh and final installment, you'll discover ways to mash all the individual bits and pieces into one autonomous and fully functioning bot.

FIGURE 1. The fully featured ArdBot with all hardware attached.



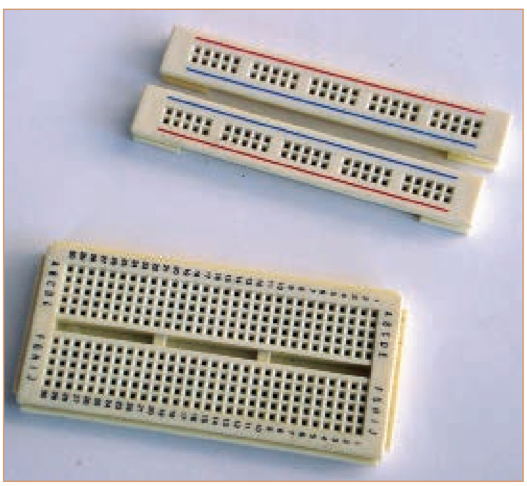


FIGURE 2. Use a 300 tie-point breadboard when constructing the fully featured ArdBot.

About the ArdBot Project

As a review, this article builds on earlier issues of *SERVO Magazine,* starting with November '10. From there, you'll find useful information about building and programming the ArdBot.

The ArdBot is shown in **Figure 1** fully populated with all sensors and accessories discussed in the previous installments. The ArdBot can be constructed out of plywood, plastic, or even picture frame mat board.

This part assumes you've read the previous articles — and ideally, built the ArdBot.

ArdBot Hardware Revisted

Up until now, the ArdBot has used a mini breadboard (with 170 tie-points) as a prototyping area. That worked when discussing individual sensors, but it doesn't offer enough room when putting everything together.

To expand the space, you can add another mini breadboard or use a 300 tie-point board like the one shown in **Figure 2**. These boards are commonly available with 400 tie-points, including two

removable contact rows. You don't need the rows, so just slip them off and save them for something else. An example is the PB-400 ($4) from All Electronics. Refer to the following illustrations for a pictorial view of wiring the solderless breadboard:

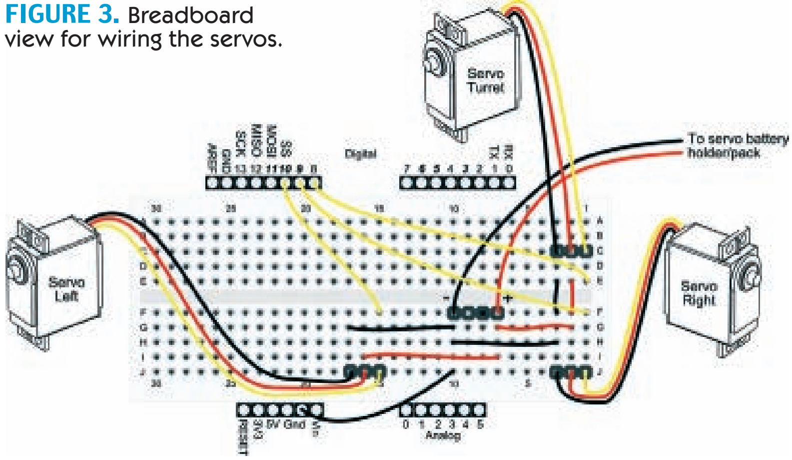
- **Figure 3** shows just the servo wiring. Note that the servos are powered by their own 4.8 to 6 volt supply.
- **Figure 4** shows the header connections and component wiring. ***Important!*** Some of the wiring may differ slightly from that in previous installments of this series, so if you're transferring parts from the mini breadboard be sure to double-check.
- **Figure 5** shows interconnections between the solderless breadboard and the Arduino.

(Feel free to use another wiring layout if you wish. Just be sure to observe correct polarity of wiring, and take care that there is only one ground connection between the Arduino and the components on the breadboard. Having multiple ground connections can cause a ground loop and erratic behavior.)

See **Figure 6** for component values and a general schematic wiring guide for the switch, photocell, and basic line following sensors. See Parts 4 and 6 of this article series for additional guidance. **Figures 7** and **8** show the wiring for the ultrasonic and Sharp infrared sensors. I'm using a Devantech SRF05 ultrasonic sensor for measuring distance, and a Sharp GP2D120 analog output infrared sensor for detecting proximity. Note the reorientation of the wiring to the Sharp sensor. This wiring follows accepted practice of placing the 5V wire in the middle. Many third-party cables designed to interface with the Sharp IR sensors use this wiring scheme (but check to be sure!).

Note: I'm using a revised connection scheme for the Sharp unit. In Part 5, I discussed how to use a transistor to selectively switch the Sharp sensor on and off. I've since determined this is counterproductive and adds more code overhead. So, you're advised to wire the power connections to the sensor directly as shown.

FIGURE 3. Breadboard view for wiring the servos.



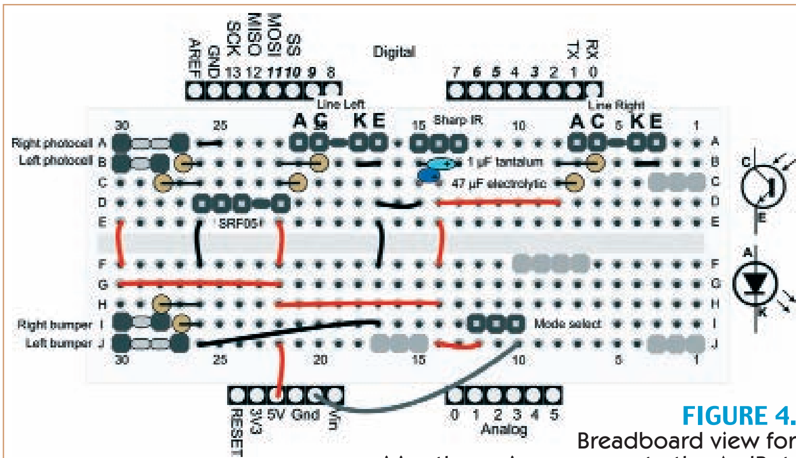


FIGURE 4. Breadboard view for wiring the various sensors to the ArdBot.

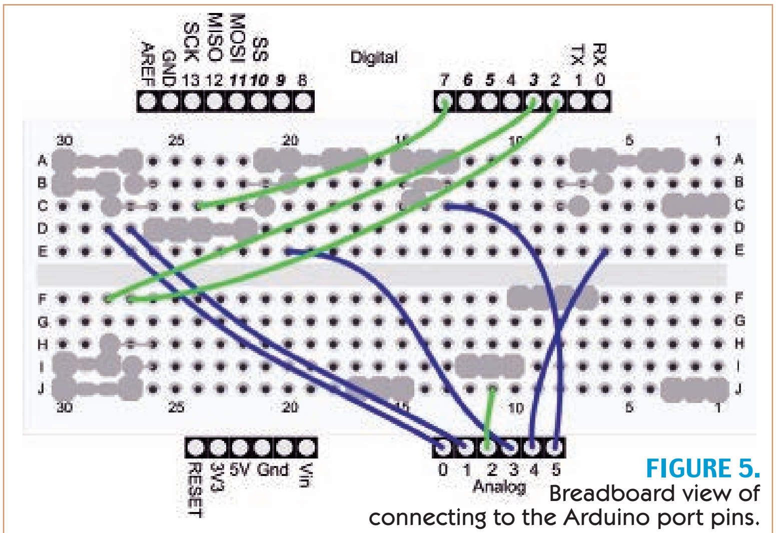
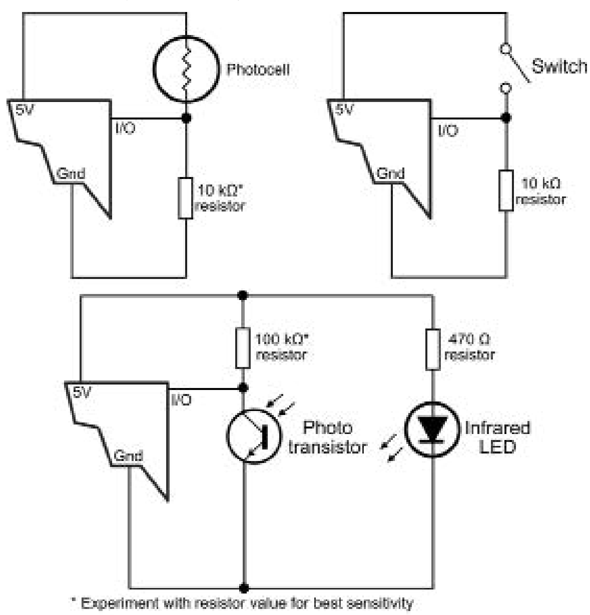


FIGURE 5. Breadboard view of connecting to the Arduino port pins.

FIGURE 6. Schematic diagram for wiring the CdS photocells, contact switches, and line following emitter/detector pairs. You need two of each.



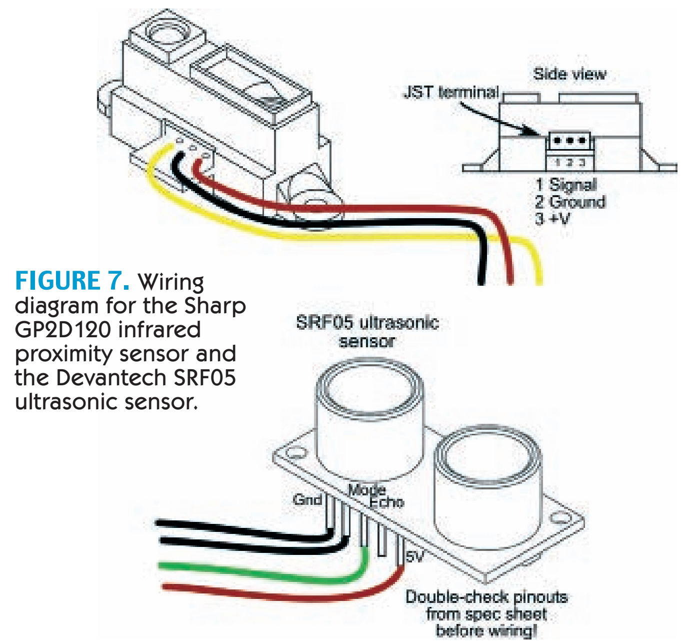


FIGURE 7. Wiring diagram for the Sharp GP2D120 infrared proximity sensor and the Devantech SRF05 ultrasonic sensor.

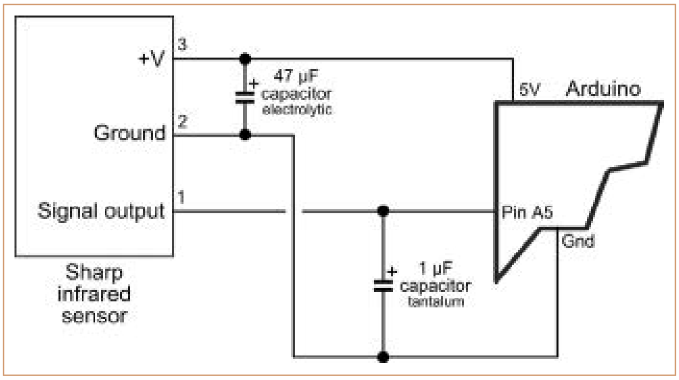


FIGURE 8. Schematic view of connecting the Sharp GP2D120 to the Arduino.

Figure 9 is the wiring guide for the optional Pololu QTR-8RC array, used as an alternative to the basic two-sensor line following optics. While the array contains eight separate infrared emitters and detectors, only five are used on the ArdBot (lack of available pins). The pin-outs are shown on the diagram. The wiring can go directly into the Arduino's digital port connectors.

Both forms of line following sensors are shown in **Figure 10**, mounted on the underside of the ArdBot. For your reference, **Figure 11** shows the pin assignments for all the components connected to the Arduino. Digital pin D13 is left free, allowing use for the Arduino's built-in LED. The serial transmit and receive pins (D0 and D1) are likewise kept free.

Powering the ArdBot

Thanks to its three servos and its complement of sensors, the ArdBot consumes a hefty amount of current. The servos operate from their own separate supply. For the servos, you can use a four-cell AA holder, with rechargeable or non-rechargeable batteries.

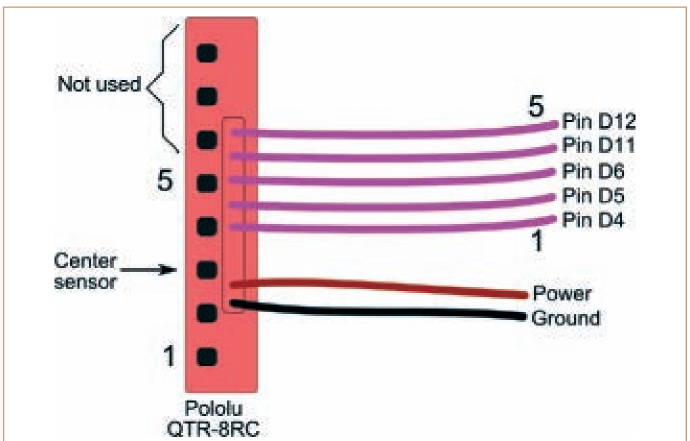
Note: Be ABSOLUTELY sure not to reverse the power and ground connections, or your servos will be INSTANTLY RUINED! As described in Part 2, the four-pin header connector on the breadboard is polarized to help prevent plugging the batteries in backward. The connector is polarized using the old broken-pin-in-the-middle trick.

The completed ArdBot contains numerous sensors that together draw over a 100 milliamps. The Arduino provides regulated five volts to these parts, so you'll want to power your Arduino from a set of AA or AAA cells, rather than the traditional nine volt battery. Use a six-cell holder or pack. Voltage with non-rechargeable cells is nine volts; it's 7.2 volts for rechargeables. Observe that the ground from the servo battery pack is electrically connected to the Arduino's *Gnd* path. This is by design, and is necessary to ensure proper operation of the Arduino and the servos. Without this common ground connection, the servos may not operate or they may behave erratically.

Programming the ArdBot

Listing 1 contains a full Arduino sketch for demonstrating the combination of hardware applied on the ArdBot. This is available in the downloads for this article. The sketch has two operating modes: *Explore* and *Line Follow*. The mode is set using a jumper block between the middle/left or middle/right pins of the Mode Select header (see **Figure 12**). Analog pin A2 is used as a digital input to read the mode setting.

FIGURE 9. Wiring diagram for the Pololu QTR-8RC line following array module to the Arduino. Find convenient connection points for the 5V and Gnd wires.



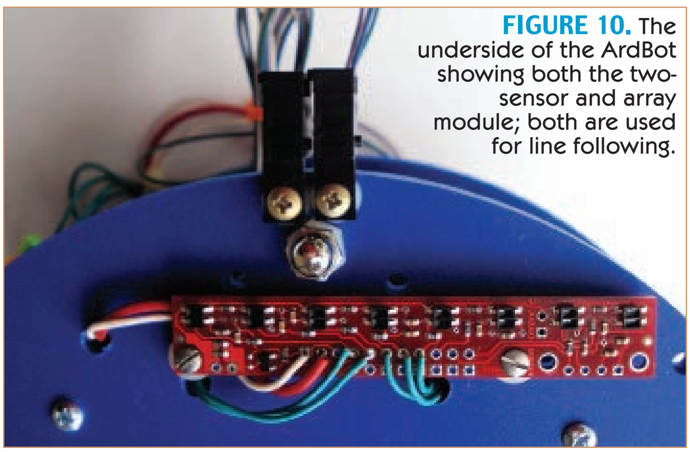


FIGURE 10. The underside of the ArdBot showing both the two-sensor and array module; both are used for line following.

• When brought LOW (grounded), the mode is set to *Explore*.
• When brought HIGH, the mode is set to *Line Follow*.

Let's dissect the sketch and discuss its most important parts. In the declarations area at the top of the sketch are references to three libraries:

```
#include <Servo.h>
#include <PololuQTRSensors.h>
#include <MsTimer2.h>
```

The *Servo* library comes with the Arduino IDE, but the other two require separate downloads. The *PololuQTRSensors* library is available from the **Pololu.com** site (see Part 6 for details) and the *MsTimer2* library is available from the Arduino page (**arduino.cc/en/Reference/Libraries**). Look under the Timing section.

As a reminder, follow these steps to add a custom library to the Arduino IDE:

1. Quit the IDE if it's running.
2. Locate the folder containing your Arduino sketches. On a Windows machine, for example, this is typically My Documents\Arduino.
3. Create a libraries folder if one doesn't already exist.
4. Unpack and place the library within the libraries folder.
5. Restart the Arduino IDE. The added library should be visible in the Sketch->Import Library menu.

The remaining lines in the declaration area are definitions for objects, constants, and variables used elsewhere in the sketch. For example,

```
Servo servoLeft;
```

creates a new Servo object called *servoLeft*. (See Parts 1 and 3 for much more on working with Servo objects.)

The required *setup()* function attaches the three servos to Arduino digital pins, specifically D8, D9, and D10. Depending on the setting of the Mode Select header, the *setup()* function then either readies the ArdBot to follow a line (digital*Read(A2) == HIGH*) or to explore.

The required *loop()* function branches off to either of two methods: *mode_line_follow* or *mode_explore*, depending on the setting of the Mode Select header.

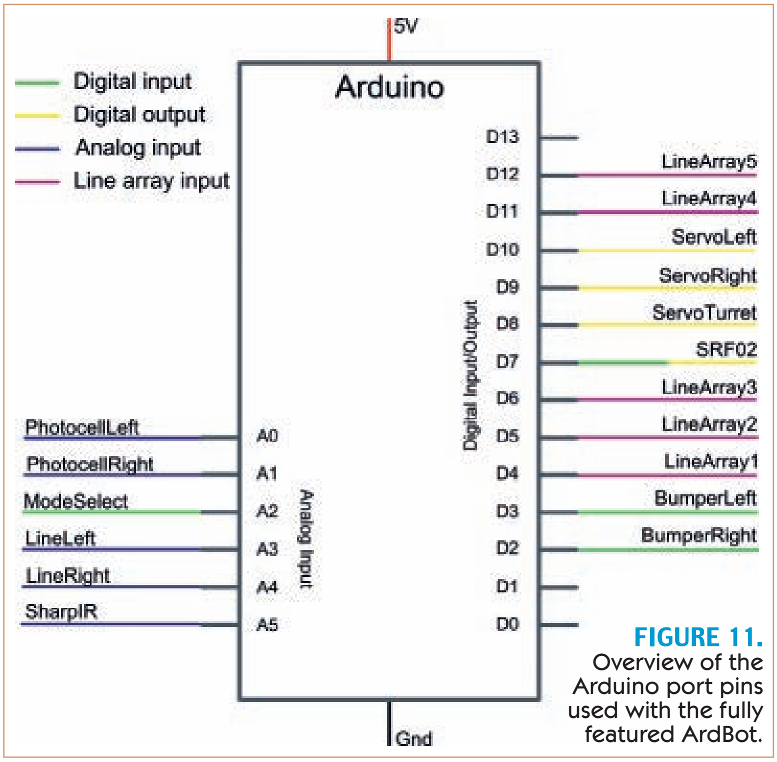


FIGURE 11. Overview of the Arduino port pins used with the fully featured ArdBot.

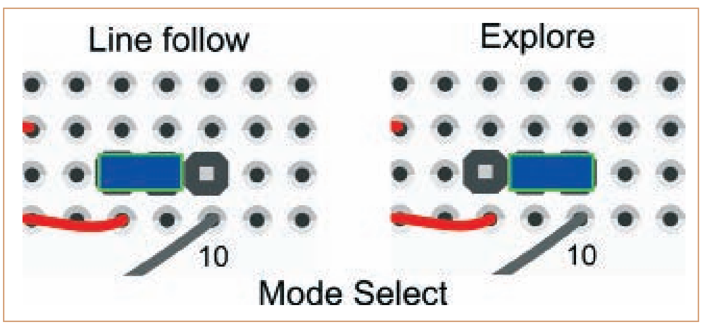
ArdBot Explore Mode

When in the Explore mode, the ArdBot wanders about a room, trying its best to stay out of trouble. It follows this routine, over and over again:

1. The robot moves forward for a predetermined period of time while monitoring its ultrasonic distance sensor for objects immediately ahead. If there's an obstacle, the bot slows down, then briefly backs up, turns, and heads off in a new direction.
2. While keeping a lookout for objects straight ahead, the ArdBot also checks the condition of its two CdS photocells. If both cells are exposed to bright light, the ArdBot responds by doing a scared side-to-side shimmy dance before heading in a new direction.
3. Every 10 seconds a special timer (provided by the *MsTimer2* library) makes the ArdBot momentarily stop and check for walls or obstacles to the immediate left or right. If there's an object close by, the bot briefly steers in the opposite direction to get away from it. The *MsTimer2* library demonstrates using one of the timers in the Arduino to trigger a hardware interrupt. More about it in a bit.
4. As an extra precaution, the ArdBot's two front bumper switches are rigged to the Arduino's external interrupt ports, pins D2 and D3. If either or both of these ports are triggered (brought from LOW to HIGH), the sketch is interrupted. The ArdBot is commanded to steer in the opposite direction of the obstacle. An object that triggers the left bumper makes the robot turn to the right, and vice versa.

Note that the Explore routine demonstrates just one of thousands of combinations you can develop using the motors and sensors on the ArdBot. Remember, that's the beauty of working with code: You can "rewire" your robot just by changing some programming lines. You're encouraged to experiment and try different things. As an aside, **Figure 13**

FIGURE 12. Use the Mode Select jumper to set the operating mode of the ArdBot to either explore a room or follow a line.



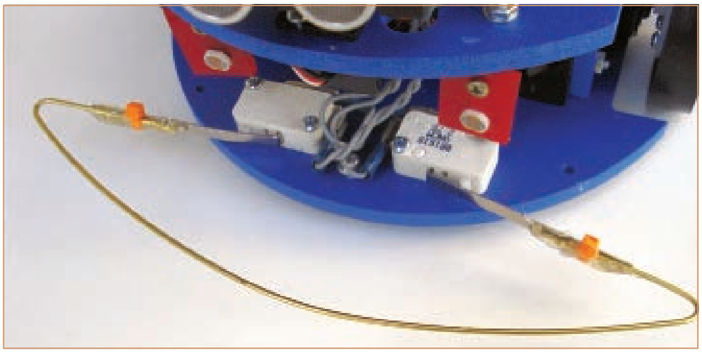


FIGURE 13. Here's one method of creating a front bumper for the ArdBot. The bumper is made with 1/16" brass rod, glued (using hot glue) to the leafs of the switches.

shows a method of attaching a 1/16" brass rod (available at hobby stores; K&S Metals stock #1626) to increase the surface area of the bumper switches. The rod is attached to the leafs of the switches using cable ties and hot melt glue.

This approach makes the switches more sensitive to contact on either side of the robot, but decreases sensitivity (due to the springy nature of the metal) to bumps directly in the middle. Feel free to experiment with different front bumper designs, including cutting 1/8" out of the middle of the rod and reinforcing the bridge with a rubber tube.

Using the MsTimer2 Interrupt

The *MsTimer2* library is a convenient method for adding a timed event to the Arduino. (There are many alternative libraries available; I picked *MsTimer2* because it's readily accessible from the Arduino website and is very simple to use.) Under the hood, *MsTimer2* uses the Arduino's eight-bit Timer2 hardware timer. This timer is a shared resource. It's available for use in your sketches as long as another library or function isn't already occupying it. Apart from the Servo library — which depends on other timer hardware — the timers on the Arduino are used for PWM (analogWrite) functionality. As the ArdBot doesn't use PWM for anything, this means Timer2 is free and clear.

To use *MsTimer2*, you need only to add its #include reference at the top of the sketch and two lines of code in the *setup()* function:

```
MsTimer2::set(10000, turret_scan);
MsTimer2::start();
```

(Note the :: double colons. This is standard C++ syntax.) The first line sets up the timer to trigger every 10,000 milliseconds (ms), or once every 10 seconds. The *turret_scan* reference indicates which function is called when the timer dings. The second line starts the timer running. Unless you use the *MsTimer2::stop* method (which we don't), the timer automatically resets itself, and triggers every 10 seconds.

When the timer is triggered, it simply sets a value contained in the *turret_scan* function (*mstimer* = true). The next time the Arduino goes through its loop (which it does many times each second), the sketch checks the value of *mstimer*. If the value is true, the robot stops and scans its sensor turret right and left. The ultrasonic and infrared sensors are on this turret.

(During this scan, only the IR sensor is read. This is simply an arbitrary design choice made to keep the length of the code listing reasonably short. In your ArdBot, you're free — and encouraged — to use both the IR and ultrasonic sensors as a means of double-checking the validity of the values from the two sensors. It's always a good idea to get a second opinion!) The timer trigger is ignored (more accurately not immediately acted upon) at various points in the sketch. This is to prevent the turret scan from occurring when it shouldn't. This is accomplished using the *turret_enable* variable. If this variable is set to true, the turret action is enabled. If false, it's skipped for that iteration of the loop.

Responding to Bright Light

Purely as a fun diversion, the ArdBot is set to avoid bright light directly ahead. When the light falling on its two variable resistor (CdS) photocells exceeds a set threshold, the robot shakes back and forth a few times, then hightails it in a different direction. This added routine is there to make things more interesting, and to provide you with another means to externally influence the behavior of the ArdBot. Recall from Part 4 that CdS photocells are variable resistors. They are connected to the Arduino using a second fixed resistor which turns their output into a varying voltage. The brighter the light, the higher the voltage. This voltage is read by two of the Arduino's analog-to-digital converter (ADC) inputs; specifically pins A0 and A1. The voltage is converted to a 10-bit binary value, ranging from zero to 1023.

Also recall from Part 4 that different photocells can have markedly different output characteristics. Selection of the fixed resistor requires some experimentation (I like to make it about the same resistance as the photocell when exposed to normal room light). Likewise, the resulting voltage output can differ. You can compensate for these variations with the line:

```
#define BRIGHT 800
```

which here stipulates that a bright light is anything that causes the ADC to read 800 or higher. If your CdS cells are more sensitive, try an even higher value; if less sensitive, try a lower value.

Reading the Ultrasonic and Infrared Sensors

The ArdBot uses both ultrasonic and Sharp infrared sensors to measure distance and proximity. While both sensors can technically be applied to either job, on the ArdBot the ultrasonic sensor is used for measuring distance, while the infrared sensor is used to determine close proximity.

The sketch uses separate functions for reading the ultrasonic and IR sensors, though they both operate in a similar fashion. Each time the function is called (*sonarRead()* for the ultrasonic sensor; *irRead()* for the infrared sensor), its value is retrieved five times. The five separate readings are then averaged out, to help avoid spurious results.

In the case of the Sharp IR sensor, each read is separated by 50 milliseconds. This represents the worst-case delay of the internal samples taken by the sensor (which, according to the datasheet, is about 38 ms, ± 10 ms). The delay is required so that each reading is from a different sample taken by the sensor.

The Devantech SRF05 sensor doesn't require so much time between each reading. Its time delay is set at just one millisecond.

Note that readings from the ultrasonic and IR sensors are not taken while the turret servo is in motion. The servo is commanded to a new position and stops to take the reading. This is to help reduce measurement errors.

Likewise, the ultrasonic sensor is not read when the robot is turning. Lateral motion while taking an ultrasonic measurement can result in wild inaccuracies, though this greatly depends on the specific sensor you're using. You get the best results when the robot is still or moving at a reasonable speed forward or backward.

As noted above, the ultrasonic sensor continually "pings" for objects straight ahead while the robot is traveling forward. If an object is detected to within three inches, the servos are slowed down (by adjusting their values close to center, which is a value of 90). If an object is detected to within an inch, the robot stops, backs up, spins for half a second to reorient itself, and then continues forward again.

Adjusting Timing Delays

The sketch relies heavily on delays to steer the robot, and to position the turret servo before taking a sensor reading. For the purpose of programming, the length of these delays is determined empirically. The delay values are based on the speed of the servos when operated from a fresh battery supply. Some servo motors are particularly fast; some are downright sluggish.

Bear in mind that servo motors slow down as the battery voltage decreases. A one second delay when the batteries are fresh might turn the robot one complete circle. But when the batteries are about dead, the same one second delay might turn the robot only half way.

In the ArdBot, most timing delays are set directly in code. An exception is the *T_DELAY* value, specified in the declaration area of the sketch. This value determines the delay used when rotating the sensor turret. The delay must be long enough to transit the servo from one extreme to the other, plus the time it takes to read the IR sensor (which is about 270 ms for all five samples).

You should strive to set delays based on average speeds. The ArdBot as presented lacks sensors for true navigation (compass, gyro, and wheel encoders for odometry), so the motion routines are intended only to demonstrate basic maneuverability. In a more elaborate ArdBot setup, you could use wheel encoders and/or a compass to more precisely determine how far the robot has turned. (It's best to use both, if you can; wheels slip, resulting in errors in the wheel encoders. The compass is used to reorient the position of the robot to a known reference.) You may also wish to explore ways to avoid using delays, as the *delay()* statement literally stops the sketch until the specified time period has elapsed. The *MsTimer2* library is a good example of using other techniques to time events. A number of ready-made libraries provide this kind of functionality, and allow you to easily create multiple timing objects.

As an example, try *Metro*, also available from the Arduino Libraries page. *Metro* allows you to create multiple timing objects, with each object set to "expire" after a certain length of time. With some crafty coding, you could replace most or all the *delay()* statements in the sketch with *Metro* timers, allowing the program to be more multi-tasking and responsive to sensor events.

ArdBot Line Following Mode

When the Mode Select header is placed so that pin A2 is brought HIGH, upon startup the ArdBot is put into line following mode. As described in Part 6, I've provided two variations of line following sensors:

1. *Basic two-sensor detector*, using a pair of infrared emitters/detectors side by side. The two detectors are meant to straddle the 3/4" line of a length of black electrical tape. Use the *mode_line_follow* method with the two-sensor detector. The sketch assumes the detectors are wired as shown in the **figures**, and are connected to Arduino pins A3 and A4.

2. *Enhanced multi-sensor array*, using (in this case) a Pololu QTR-8RC eight-detector array. These units are very compact and cost less than when purchasing separate ready-made infrared emitter/detector modules. Use the *mode_line_follow_array* method with the QTR-8RC. The ArdBot uses five of the eight available emitters/detectors, and these are connected to digital pins D4, D5, D6, D11, and D12.

Listing 1 includes the code for both line following variations. When using an array — like the QTR-8RC — it is also possible to detect a "finish line" in the line following course. See the ArdBot Construction Notes site (check the **Sources** box) for more information.

For More Arduino Coverage

Though this is the final article in this series, there is plenty more to share about combining the Arduino microcontroller with robots. After all, there's so much more to talk about. So, stay tuned! Be sure to check out the ArdBot Project updates, free code, and additional examples at **www.robotoid.com/servomag/**. **SV**

Sources

If you'd like to build the ArdBot, be sure to start with the November '10 issue of *SERVO Magazine* for Part 1 of this series. Also check out the following sources for parts:

Prefabricated ArdBot body pieces with all construction hardware:

Sensor Turret Budget Robotics
www.budgetrobotics.com

ArdBot Construction Notes:
Downloadable code, updates, additional sketches, and examples.
www.robotoid.com/servomag

Arduino Resources:

Arduino
www.arduino.cc

Fritzing
www.fritzing.org

Online Retailers of Arduino, infrared sensors for line following, and components:

AdaFruit Industries
www.adafruit.com

All Electronics
www.allelectronics.com

Digi-Key
www.digikey.com

Jameco
www.jameco.com

Lynxmotion
www.lynxmotion.com

Parallax
www.parallax.com

Pololu Robotics & Electronics
www.pololu.com

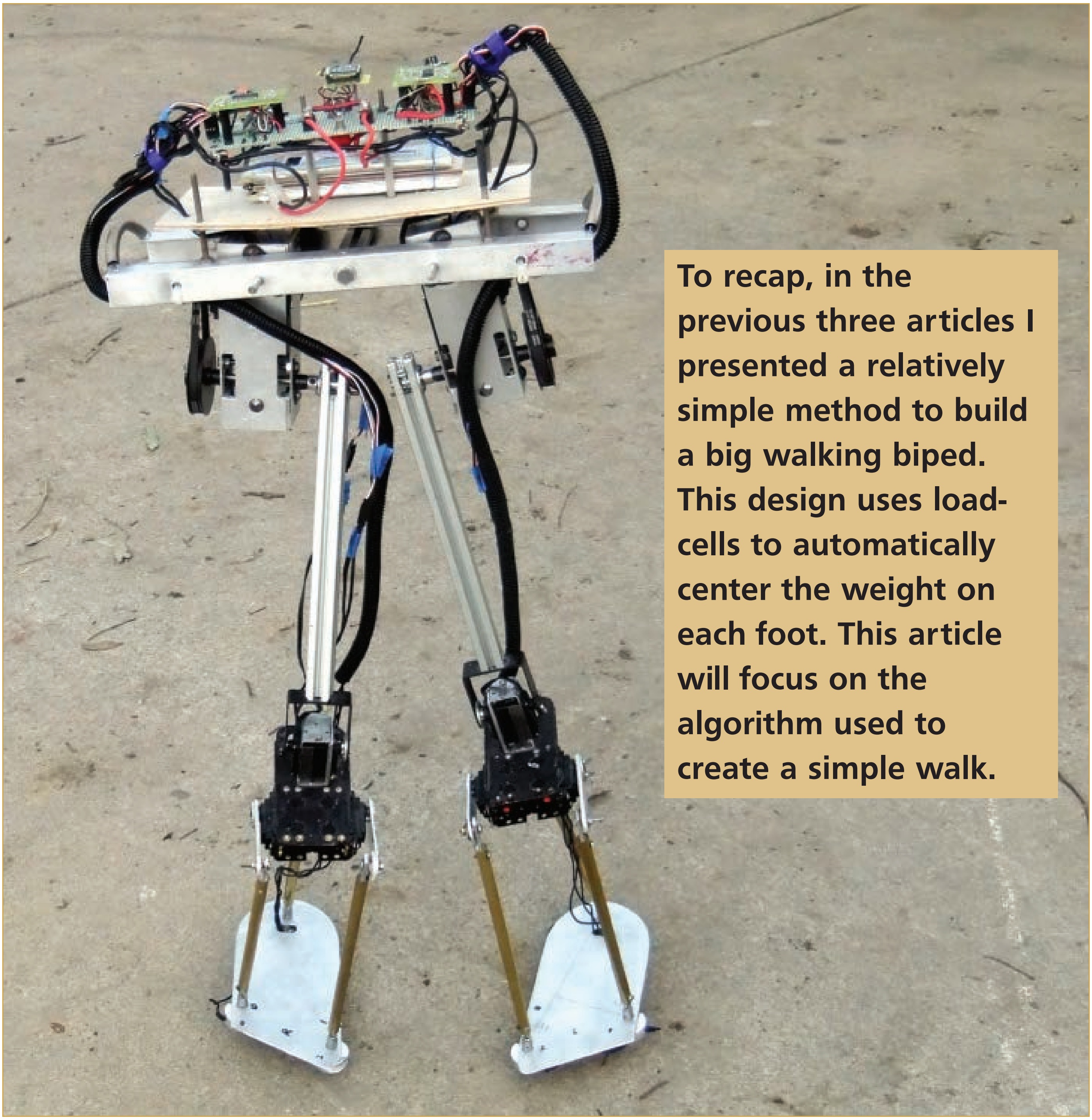
RobotShop
www.robotshop.com

SparkFun
www.sparkfun.com

Build Your Own Big Walker Part 4

by Daniel Albert

www.servomagazine.com/index.php?/magazine/article/may2011_Albert



To recap, in the previous three articles I presented a relatively simple method to build a big walking biped. This design uses load-cells to automatically center the weight on each foot. This article will focus on the algorithm used to create a simple walk.

All of the bipeds I have previously worked on — the KHR-1, the RoboNova, and my own custom-built 18 inch biped — used a static frame model. The static model creates motion similar to a movie. Each frame is individually defined as a non-moving stable position. Transitioning from one frame to another in a sequence creates movement. This is a very easy method to create a walking robot. These robots usually have no idea if the floor is level or smooth. A sequence that walks on a five-degree upward incline may not work on the downward or sideways incline. Several engineering fixes like really big feet, gyros, and accelerometers have been added to overcome this limitation. Some people have had moderate success with these approaches. To get really good results using gyros and accelerometers takes a great deal of programming know-how. At the minimum, you should know how to program and tune a Kalman filter to merge the gyro and accelerometer data. One of the features and downfalls of the static model is that at each frame, the robot is balanced and the center of gravity is directly under the mass of the robot. People do not walk this way. We throw our weight away from the center and catch ourselves with our descending foot. This is the dynamic way of walking.

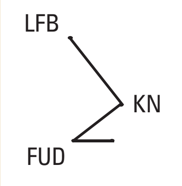
This method is extremely difficult to program. Many P.h.Ds have written theoretical papers on different dynamic methods. Few have built real working models. What really makes me mad is that some have even filed successful patents on dynamic walking methods without being able to build a working model. I don't have the math background to design such a system. So, without "throwing the baby out with the bath water" I came up with a hybrid dynamic/static model that was simple to implement and simple to program.

What is hybrid about this system is that from the knees up, the servos are driven by static frames and the ankle and foot are equipped with dynamic compensators. These allow walking on uneven floors and help with unruly servos and slop in the mechanics. The feet servos are dynamically compensated by the center of gravity algorithm discussed in previous articles. The big walker currently only has a lower body. The lower section of a typical biped has at least 10 degrees of freedom (i.e., angles that vary). So does the big walker.

As is standard, there is one servo for each degree of freedom. Note from Part 2 that while the foot uses two servos for two degrees of freedom, both servos contribute 50/50 to each degree of freedom. So, in total, each leg requires two for the hip, one for the knee, and two for the feet. The program uses static settings for the feet only to give a good starting place for the foot position. This is easy to calculate. Picture a static frame on a level surface with a leg that is positioned at a 10 degree angle out away from the hip; the foot will need to lean in 10 degrees to keep the foot parallel to the floor, and thus the weight is balanced to the middle of the foot. If the floor suffers any deviation from level, then the weight is not centered and the robot is less stable. With the hybrid system, the load-cells adjust the position of the foot so we can be less concerned with coordinating the angles of the hip and knee. Once one of the three cells touches the floor, the foot will auto-adjust the three points until the weight is centered. The correction loop runs at 20 times a second. **Table 1** shows some nomenclature used to help describe the servos of the leg. LFB, KN, and FUD form a triangle. For convenience, I have set the orientation of the servos such that LFB + KN + the complimentary angle of FUD will add up to 180 degrees. On a level surface, the hips are parallel. This is for calculation of the static frame only. FUD will always auto-adjust to center the mass at the center of the foot. Likewise, for convenience, the LIO and the FIO will add up to zero on a level surface and level hips. Again, FIO will auto-adjust to center the mass.

TABLE 1.

Servo	Description
LIO	Leg In and Out from Hip
LFB	Leg Forward and Backward from Hip
KN	Knee
FIO	Foot Lean Sideways In and Out
FUD	Foot (Toe) Up and Down



The Movement

A good place to start any movement is from a stable standing position. It is also a good place to end up after completing a series of frames. This way, with a common frame in each sequence you can easily transition to another upon completion. For a simple walk, we need only a small number of frames. The frames will specify angles for all five servos of each leg. Frames can have the auto-balance feature turned on or off. I found that in the initial stand it is better to not have the auto-balance on. Even in the auto-balance mode, the two foot servo positions are maintained statically when there is no weight on the foot. Since the biped is symmetrical, each LPB (Limb Processing Board) has its own "footedness" opposite from the other, right or left. Frames are designed

TABLE 2.

	Left/Right Leg						Right/Left Leg				
Frame Name	LIO	FIO	KN	LFB	FUD	Bal	LIO	FIO	KN	LFB	FUD
TWOfootSTAND	10	-10	170	5	5	Off	10	-10	170	5	5
LEAN	-5	5	176	2	2	Auto	25	-25	160	10	10
LEG_FWD	-5	5	160	10	10	Auto	25	-25	170	10	5
PLANT_FOOT	-5	5	172	4	4	Auto	25	-25	158	11	11
SHIFThipFWD	25	-25	160	10	10	Auto	-10	10	170	10	10

to be flipped from left to right, i.e., left-leg-forward and right-leg-forward can be the same frame but footedness flipped. (I wasn't sure until I looked it up, but footedness is a real word!) Check out **Table 2**. I initialize the footedness as "right." The sequence for a walk is as follows:

1. TWOfootSTAND.
2. LEAN (one leg will be off the ground).
3. LEG_FWD(one leg still off the ground).
4. PLANT_FOOT (partial weight on planted foot).
5. SHIFThipFWD (other leg is off the ground).
6. Change footedness.
7. Go back to step 3.

The sequence itself is sent over the ZigBee link from a PC program that holds the frame list. The master LPB sends the servo positions and load-cell data for both LPBs back to the P/C. The command list is quite simple. Multiple commands can be concatenated via a comma. A CR(0xd) is the delimiter that causes execution of the commands sent. See **Table 3**.

TABLE 3.

Command	Parameter	Param Name	Param Limits	Parameter Description
$	n	LPB ID	0 - 255	0 = master, 1 = slave
#	n	Servo Number	0 - 255	One of the five servos
P	n	Position	500 - 1500	500 = -90 degrees; 1500 = +90
F	a	Footedness	R or L	Right or Left
C	a	Auto	A or "	A turns on auto-centering. Space turns off.
T	n	Sensor #	0,1,2 or 3	0 = tare all sensors; 1, 2, or 3 individually tare sensors.

The Results

I designed the robot for fast movement. It isn't. To keep it from falling over, I need to move really slow. The biggest problem is still slop in the mechanics. Oscillations start to creep in as the robot moves and after several steps, it becomes uncontrollable. However, all and all this first version was very successful. Without the load-cell based dynamics in the feet, it couldn't walk at all. So, I will take the two inch baby steps as positive proof that this system can work.

The Next Step … Pun Intended

I plan on replacing several of the joints with stiffer rod-ends. Springs (like ligaments) will be added to hold joints in place and reduce oscillations. The firmware needs an additional PI (Proportional Integral) loop to reduce overshoot. I plan to have these features added in the coming months and look forward to reporting even better success. **SV**

CPLDs — Part 3

complex programmable logic devices

Simulating a Digital Design

by David A. Ward

After reading the first two articles in this series of five on CPLDs, you should be able to enter a digital logic circuit into the Xilinx software through the graphical or schematic entry method, compile that design, and program the configuration into a CPLD. This article will demonstrate how to simulate your digital logic design in Xilinx Isim.

www.servomagazine.com/index.php?/magazine/article/may2011_Ward

To begin with, enter the circuit you want to simulate into the Xilinx software. We'll use a simple two-gate circuit with three inputs and one output (see **Figure 1**). We won't go into all of the details on how this was done; you can refer back to the second article if you need to. We now need to generate an HDL test bench that will describe the signals we'd like input into our circuit for simulation purposes. Select Project>New Source from the top menu as shown in **Figure 2**. The next window asks which type of file we want to add to the design; select VHDL Test Bench (see **Figure 3**). It's also a good idea to add the letters TB into the file name so that whenever the file name appears, you can quickly differentiate it from other types of files. **Figure 4** shows you which source file the test bench file will be associated with; in some designs, you may have several source files open at one time. In this demonstration, only one is open; select Next. The next window that will open is the summary window shown in **Figure 6**; select Finish. **Figure 6** shows the HDL test bench template that was prepared for you. Notice in **Figure 7** that an area is marked between the two green comment lines. That's the user defined section where we will enter our test signal information into. **Figure 8** shows the information that we will

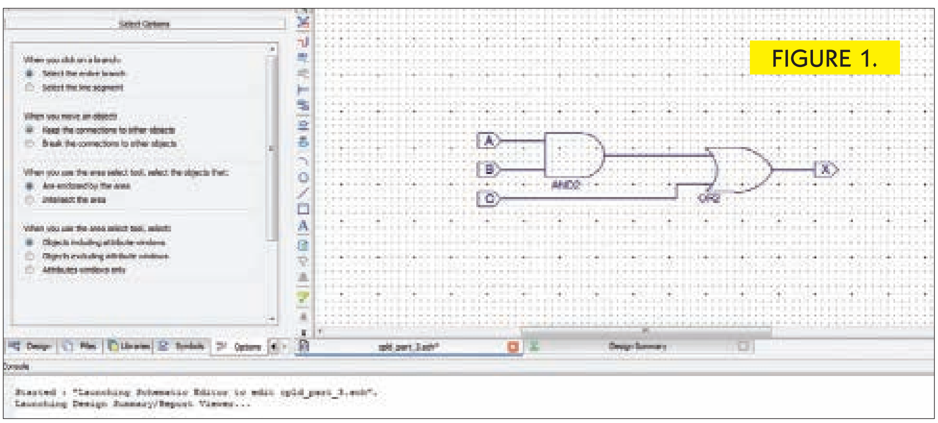


FIGURE 1.

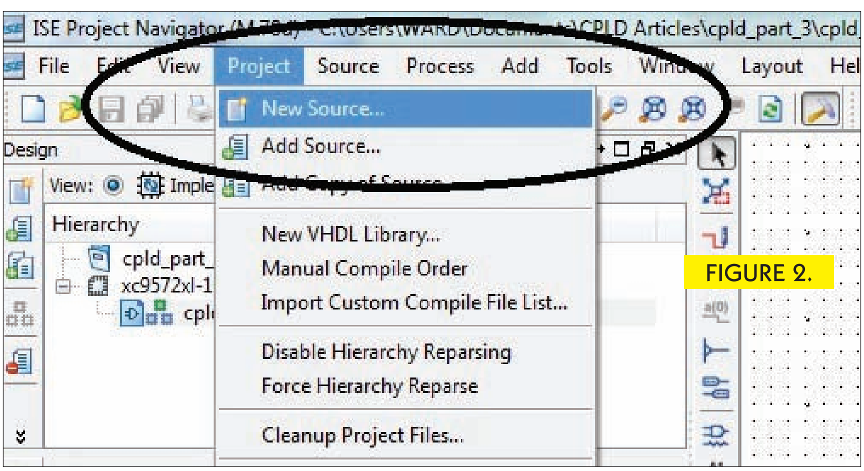


FIGURE 2.

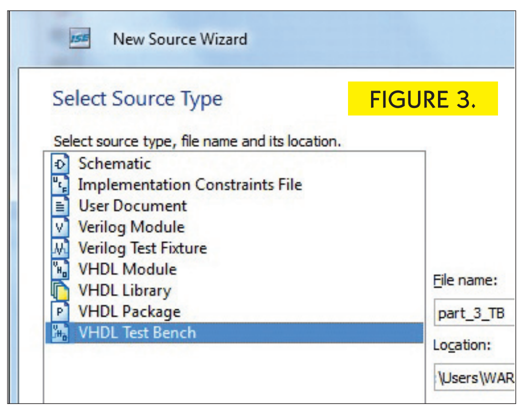


FIGURE 3.

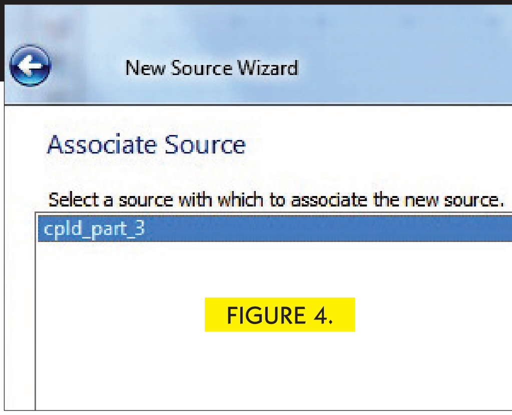


FIGURE 4.

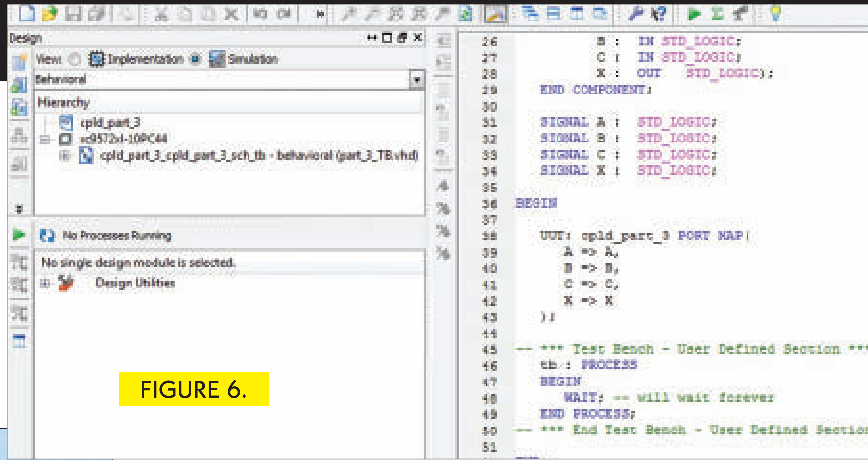


FIGURE 6.

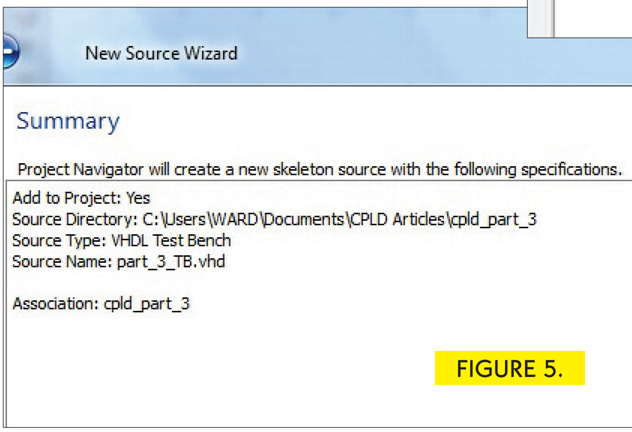


FIGURE 5.

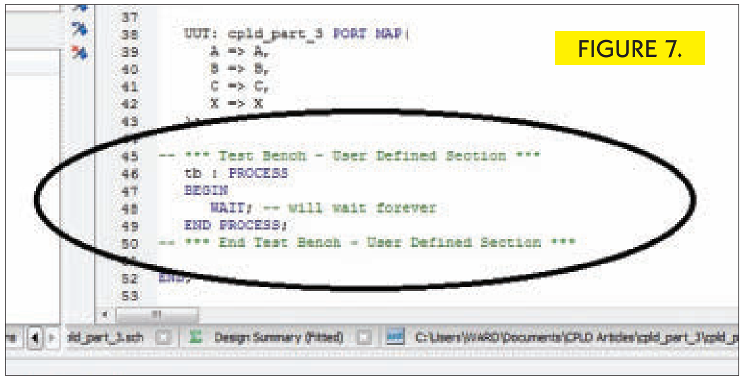


FIGURE 7.

use for our circuit.

Let's take a closer look at the three tb_ processes to see what is going to occur during the simulation. The first process, tb_A, will place a signal onto pin A of our circuit and make it a '0' for 500 nS, and then a '1' for 500 nS. This will be repeated over and over again when the simulation is run. This will place a 500 nS + 500 nS = 1µS period, or 1 MHz digital waveform, onto input A.

The next process, tb_B, will do the same to input B except the frequency will be half the frequency of input A (500 kHz) since its period is twice that of the signal into A.

The third process, tb_C, will do the same to input C but its frequency will be half that of input B (250 kHz), since its period is twice that of the signal into B. By injecting signals in this manner, the simulation will count in binary, in order, through all eight of the possible combinations; in this case, in eight half cycles of the signal placed into input A, or 4 µS. If your circuit has more inputs, simply double the period of each additional signal which will halve the frequency on each subsequent input that the circuit has.

Notice the placement of semicolons in the process lines. If any are missing, the test bench will not compile. Notice also the use of two dashes ("- -") to denote comment lines in HDL.

To run the simulation, select the Simulation view radio button and then Simulate Behavioral Model; then select run (see **Figure 9**). You will then be asked if you want to save the changes you have made to the test bench file (**Figure 10**); select yes. After the program works for a while, you will be taken to the Isim program as shown in **Figure 11**.

The simulation has already run with a default time setting of 1 µS, this needs to be changed for our particular

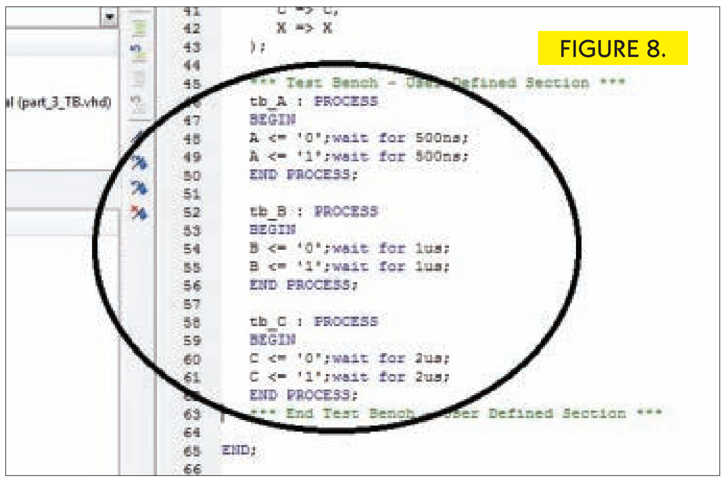


FIGURE 8.

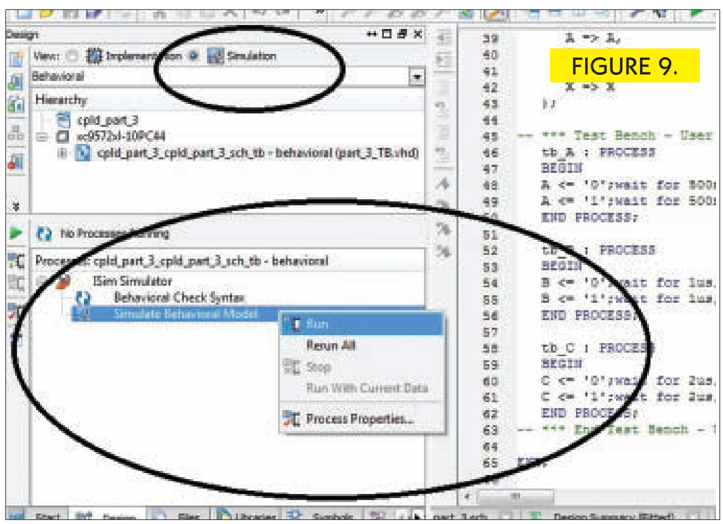


FIGURE 9.

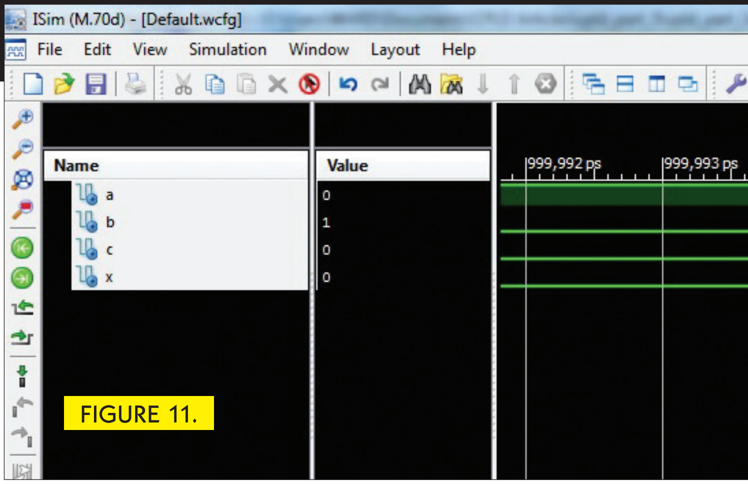


FIGURE 11.

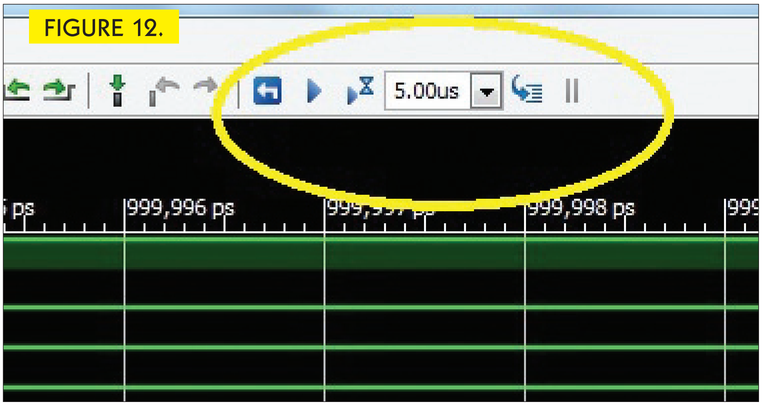


FIGURE 12.

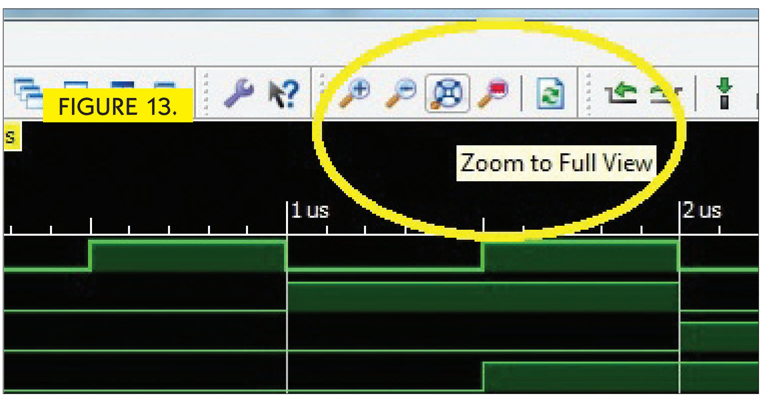


FIGURE 13.

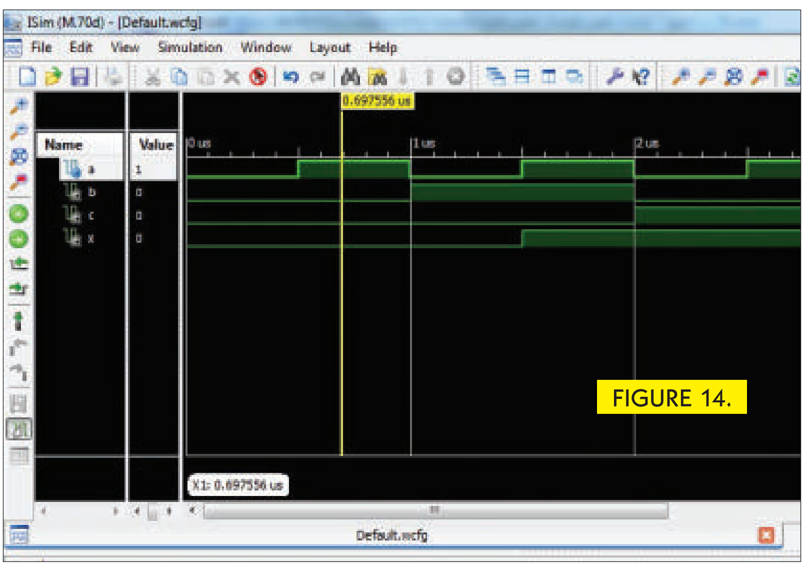


FIGURE 14.

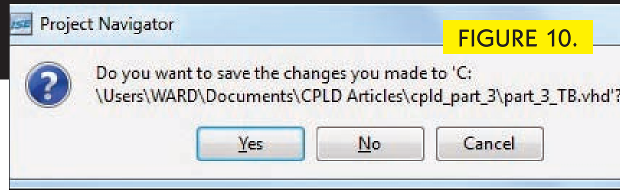


FIGURE 10.

circuit to 5 µS. From the top menu bar, change the runtime from the default time to 5 µS, then press the reset button to the left of the simulation time box. Next, select the run icon with the timer (hour glass) icon next to it to run the simulation for the 5 µS (see **Figure 12**). Now select the "zoom to full view button". It looks like a magnifier with an "X" in the center. To see your complete simulation, look at **Figure 13**.

The zoom to full view button is probably one of the most important controls you will use in the Isim program. Any time your waveforms don't look right, try clicking on the zoom button and see if that fixes the problem. By left-clicking on a waveform, you can get a cursor that will show you exactly where you are in the simulation time (see **Figure 14**). Note also on the left side of the screen that the logic values of all of the signals will be shown where the cursor was placed. You can change the colors of any of the traces by left-clicking on a trace, right-clicking for a pull-down menu, then selecting signal color, and changing the color on the color palette, as in **Figure 15**.

If you do not like how the program placed any of your traces, simply click on them and drag them up or down to a new location. The Isim simulator is a great way to see how your circuit will actually behave before you program it into a CPLD. Of course, the Isim program will do many more things than have been shown here; this was just enough information to get you started.

Let's leave the Isim program at this time and go back into the Xilinx project navigator. Last month, it was mentioned that pins can be pre-assigned and locked so they will not change positions from one compilation to the next. Xilinx recommends, however, that you let the program do the pin assigning since it will probably choose a more efficient design than you would. However, pin locking is something that you'll want to do so that you do not need to constantly change your breadboard or PCB when you make minor changes to a design.

To lock the pins, you will first need to generate a "UCF" or user constraint file. To do this, select Floorplan IO-Pre-Synthesis and run (see **Figure 16**). Next, you will see the window in **Figure 17**; select yes. This will generate the user constraint file. Next, you will be taken into the Pace program, as shown in **Figure 18**. This is where you can pre-assign pins by dragging the input and output names from the left over to the right and dropping them onto a particular pin shown in the package view. We will not pre-assign any pins at this time.

We only need to enter into Pace to generate a user constraint file so we can lock the pins. Exit

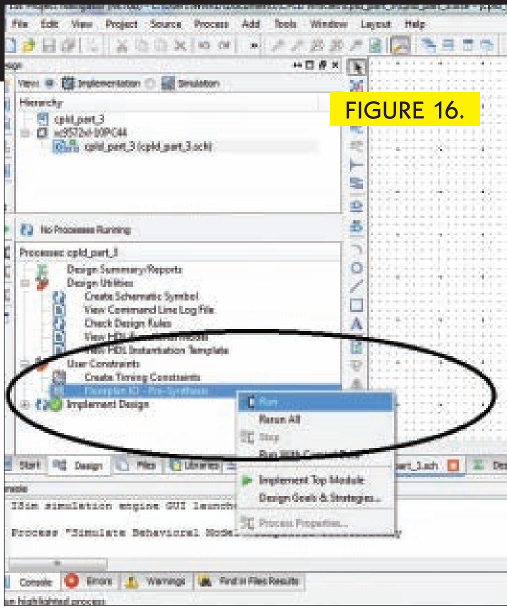


FIGURE 16.

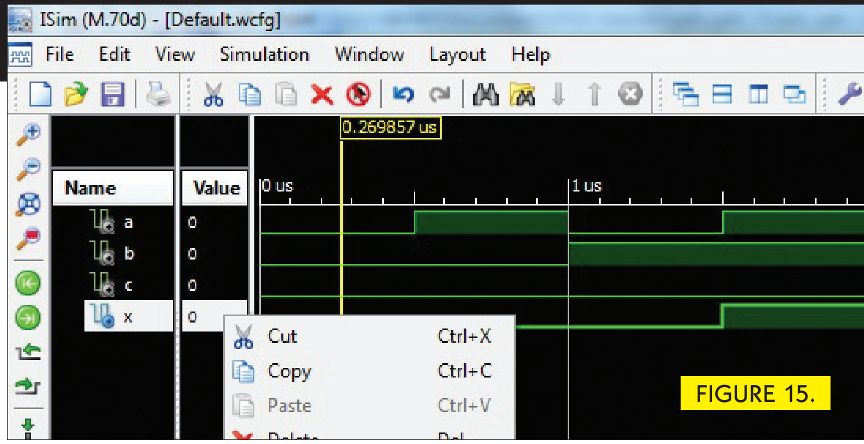


FIGURE 15.

FIGURE 17.

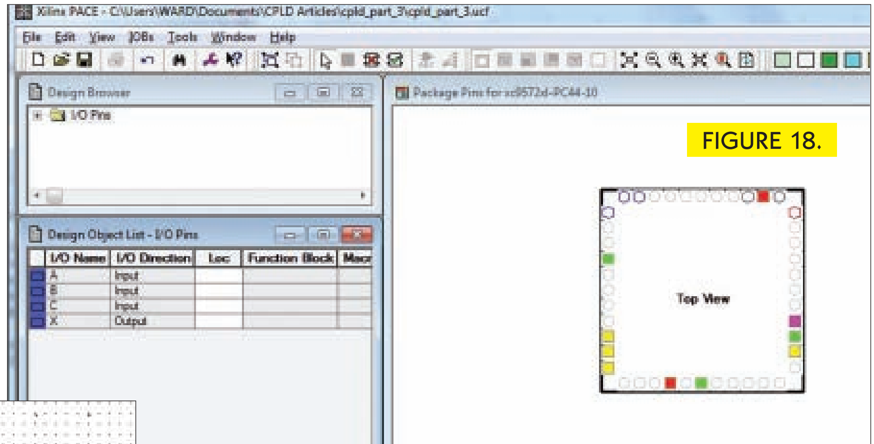


FIGURE 18.

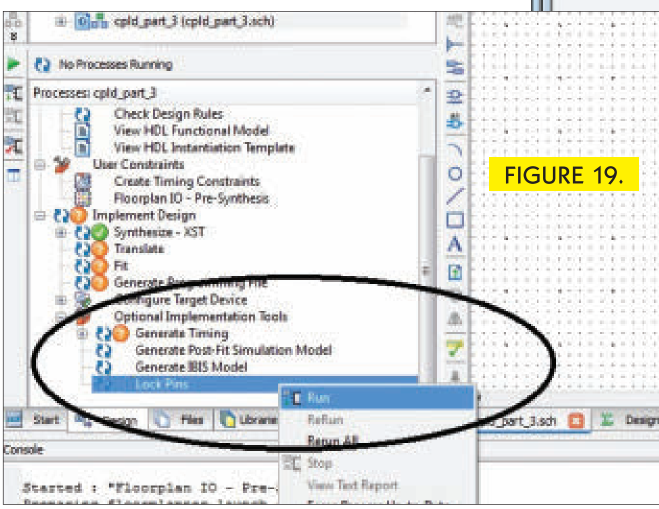


FIGURE 19.

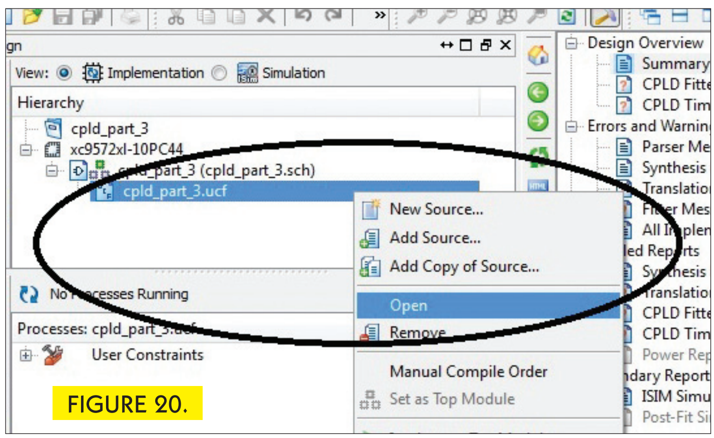


FIGURE 20.

Pace and return to the Project Navigator window. Under the optional implementation tools, select Lock Pins and run (see **Figure 19**). From here, you can open up and look at the user constraint file by left-clicking on the ucf file then right-clicking, and from the drop-down menu select open; see **Figure 20**. **Figure 21** shows what the user constraint file looks like. If, for some reason, you do not want the pins locked you can remove the user constraint file by left-clicking on the ucf file, then right-clicking for a drop-down menu; select remove.

Wrap-Up

The next article will introduce HDL programming of a CPLD; the fifth (and final) article will put all we have learned about CPLDs into a complete project design. **SV**

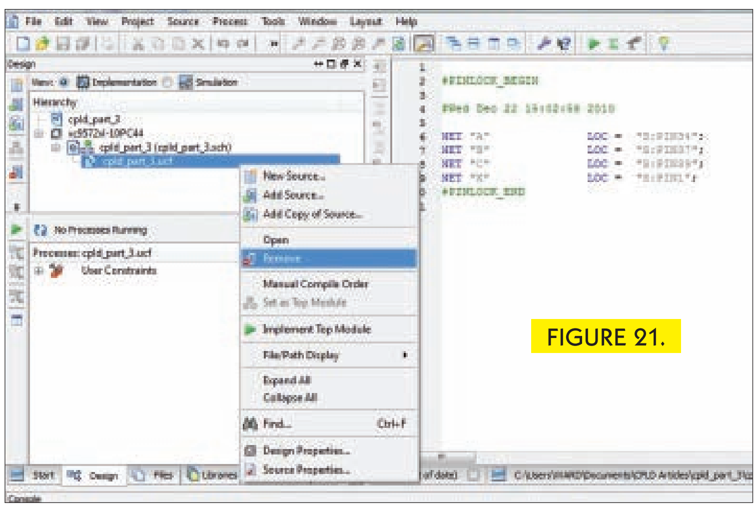


FIGURE 21.

The NXT Big Thing #10

HiTech Fetch!

By Greg Intermaggio

What's crackin, roboteers? In last month's edition of The NXT Big Thing, we used HiTechnic's compass sensor combined with the standard ultrasonic sensor to make a robot that can head due North while avoiding obstacles.

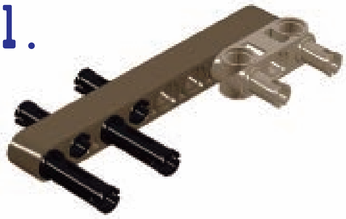
This month, we'll be exploring yet another sweet sensor from HiTechnic: the infrared (IR) Seeker sensor V2. You'll need an IR Seeker V2 and a HiTechnic infrared electronic ball — both of which can be found at **HiTechnic.com**.

This article will assume that you've already purchased the IR ball and an IR Seeker V2 sensor, and downloaded and imported the programming block into the NXT software. Download the block from HiTechnic's website and import it by clicking Tools > Block Import and Export Wizard.

All that said, let's get to our project!

Building Instructions: IR Ultrasonic Attachment

1.

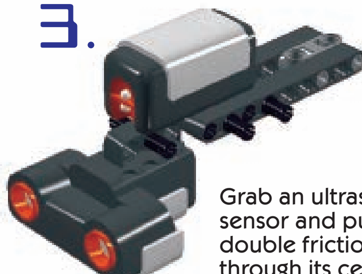


Start with a nine-hole studless beam with the pins indicated attached.

2.

Snap in your HiTechnic IR Seeker sensor. (Note: The one pictured is a light sensor. Your IR Seeker sensor will have a black plastic face.) Then, snap in another nine-hole studless beam with two friction pins on each side.

3.

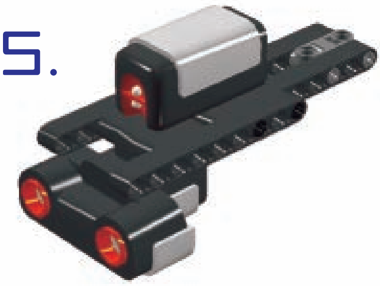


Grab an ultrasonic sensor and push a double friction pin through its center hole.

4.

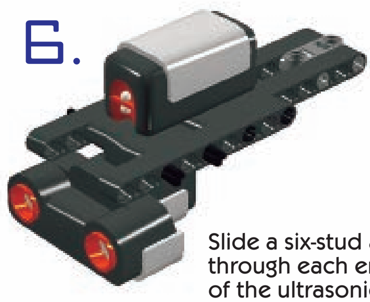
Attach a three-hole studless beam to each side of the ultrasonic sensor.

5.



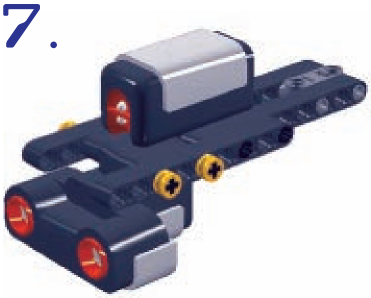
Snap nine-hole studless beams into the pins on the assembly with the IR sensor. Align them with the ultrasonic sensor as shown (they won't be connected yet to the ultrasonic sensor).

6.



Slide a six-stud axle through each end of the ultrasonic sensor's holes to begin attaching it to the rest of the assembly.

7.

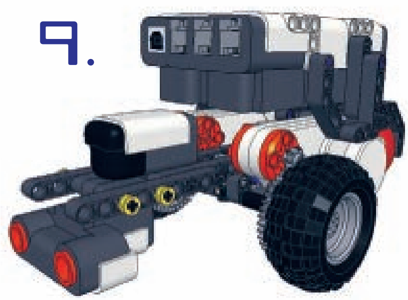


Close off those axles with half-bushings.

8.

This is how your IR/ultrasonic attachment should look with the HiTechnic sensor attached.

9.



Snap the assembly to Eddie 2 the same way we've done in the past, and you're good to go!

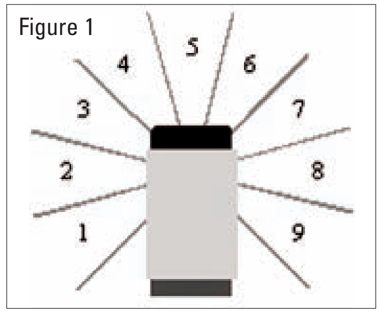
Plug the IR sensor into port 1 and the ultrasonic sensor into port 4.

Testing the IR Sensor

The IR Seeker sensor detects the position of the IR ball relative to the sensor. If the ball is straight ahead, the sensor outputs a value of 5. If the ball's a bit to the left, it outputs a 4, and if it's a bit more to the left, a 3. Likewise, if the ball is to the right of the sensor, the value output from the sensor will increase.

Figure 1 shows the approximate outputs of the IR Seeker sensor when the ball is in various positions. One is the lowest possible output and nine is the highest possible output. The first thing we want to do is test the sensor, so let's write a simple program to show us its output. Now, run your program and turn on the IR emitting ball. You should see a number displayed on the NXT's screen. That number should be somewhere between one and nine. It should decrease as the ball moves further left relative to the sensor and increase as the ball moves further right relative to the sensor.

Once you've got this program working, let's move on to the main event



IR Test Program Instructions

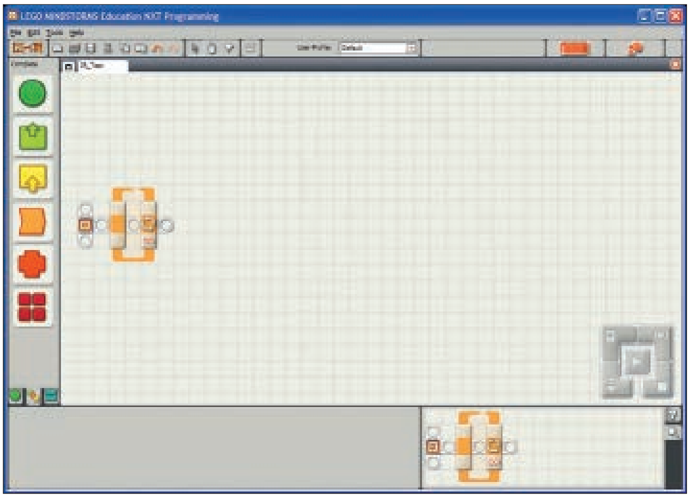


Figure 1. Start with a loop.

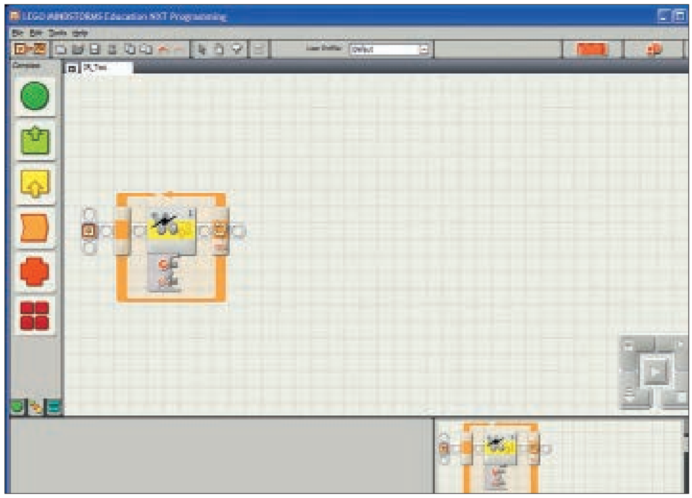


Figure 2. Add an IR Sensor block (see the introduction for instructions on how to import this block if you haven't already).

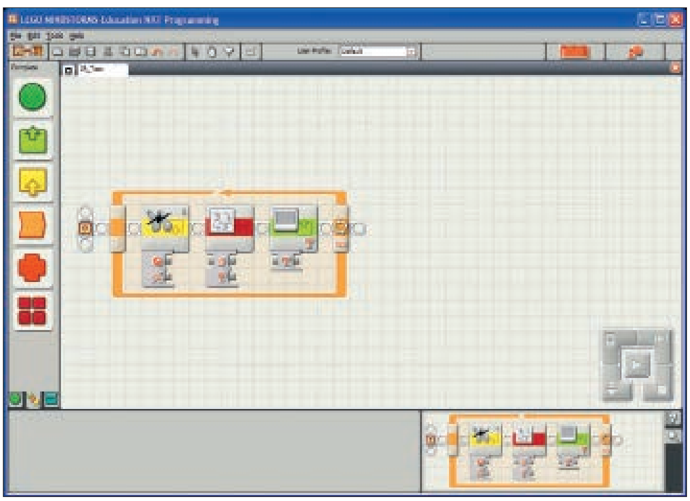


Figure 3. From the Advanced tab, add a "Number to Text" block, then add a display block to Text.

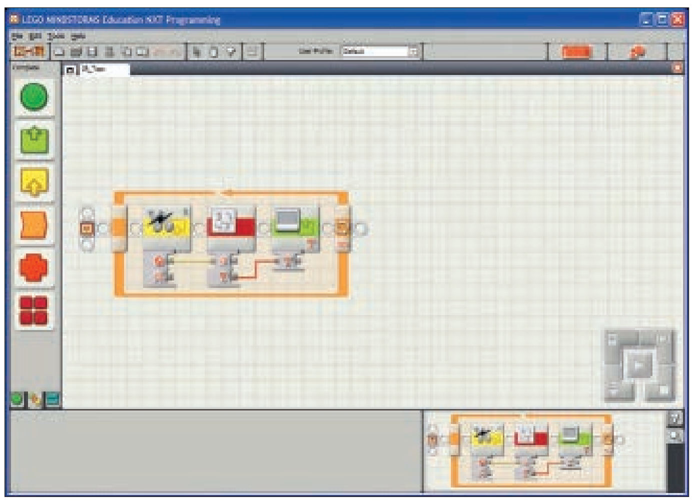


Figure 4. Wire the IRDirection data hub on the IR Sensor block to the Number data hub on the Number to Text block. Then, wire the Text data hub on the Number to Text block to the Text data hub on the Display block.

Seeking The Ball

We're going to program Eddie to follow the infrared ball until he's right up next to it. Let's think for a minute about what we know:

- Eddie is going to need to adjust his steering dynamically, based on his position relative to the IR ball.

- Steering can be controlled by inputting a number between -100 and 100 (-100 is a sharp left turn, 0 is straight forward, 100 is a sharp right turn).
- The IR Seeker sensor outputs a value between one and nine (1 when the ball is far left of the sensor, 5 when it's dead ahead, and 9 when it's far right).

See that? We need to change 1 to -100, 5 to 0, and 9 to 100. So, how do we do it?

First, let's subtract five from the IR sensor value. That makes a reading of 5 (ball is dead ahead) turn into a steering of 0 (go straight forward). If you subtract that same five from a sensor reading of 1 (ball to the left of the sensor), that leaves us with -4. Likewise, subtracting the same five from a reading of 9 (ball to the right of the sensor) leaves us with a 4. We want to turn that -4 into a -100 for a full left turn, and turn that 4 into a 100 for a full right turn.

$-4(x) = -100$
$4(x) = 100$

Four times what is 100? 25! And -4 times what is -100? Also 25! So, all we need to do after subtracting five from the sensor reading is multiply the result by four. This won't be so complicated after all! Let's get to programming.

Download and run your program, and turn on the IR ball. Eddie should now roll towards the ball. When he reaches it, he'll stop and make whichever noise you've selected.

Ball Program Instructions

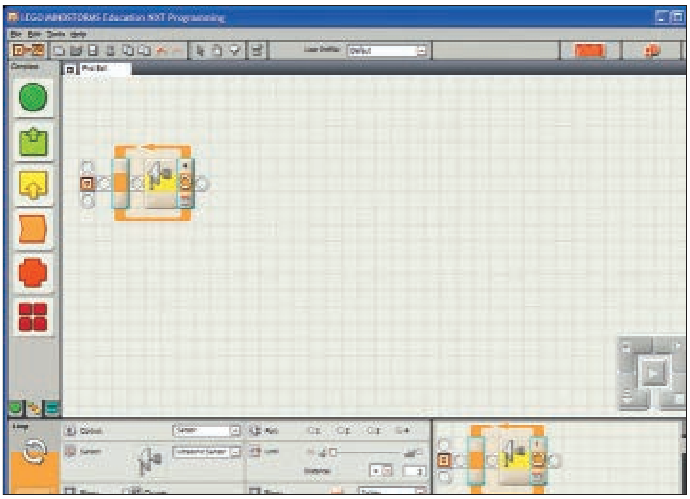


Figure 1. Start with a loop. Set the Control to Sensor, and the Sensor to Ultrasonic Sensor. Set the Sensor Port to 4 and the distance to less than (<) 3 inches.

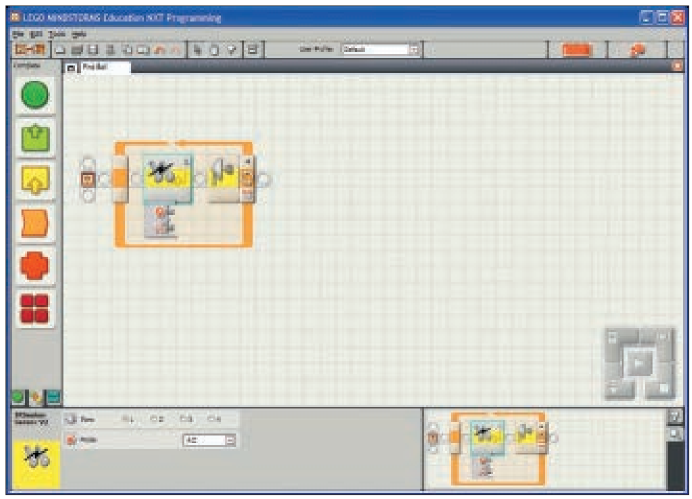


Figure 2. Add an IR Sensor block set to port 1.

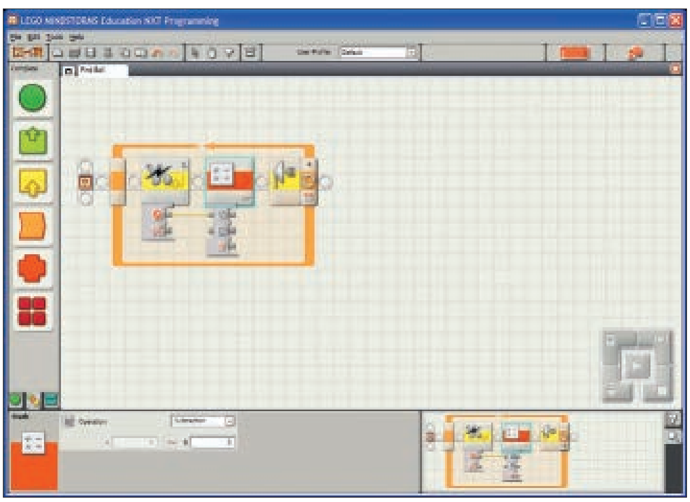


Figure 3. Add a Math block from the Data tab. Set the operation to Subtraction. Wire the IRDirection data hub on the IR Sensor block to the A Data hub on the Math block.

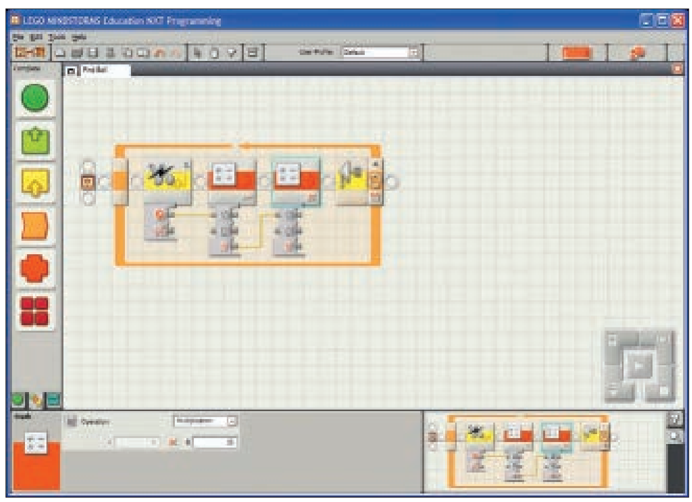


Figure 4. Add a second Math block. Set the operation of this one to multiplication. Attach the Result from the first Math block to the A Data hub of this one, and set B to 25.

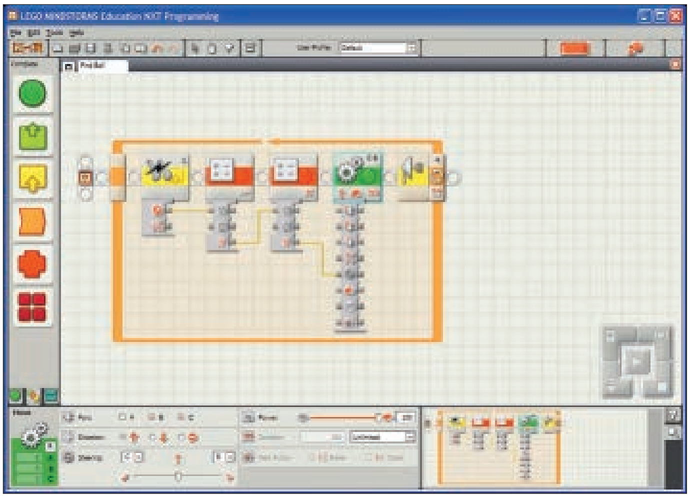


Figure 5. Add a Move block. Set the power to 100 and the duration to Unlimited. Wire the Result data hub from the last Math block to the Steering data hub of the Move block.

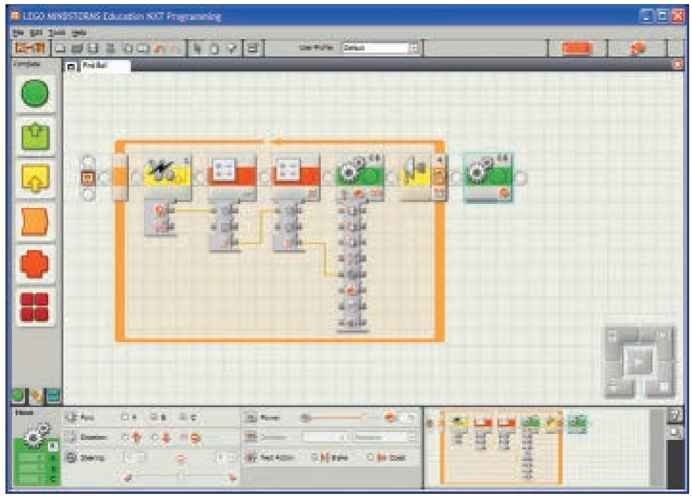


Figure 6. Add a Move block outside of the ultrasonic sensor loop, and set the direction to Stop.

Recap

In this edition of The NXT Big Thing, we used an Infrared Seeker sensor to make Eddie chase after an IR ball, and an ultrasonic sensor to detect when he reaches it. Check back next month for our next exciting installment! **SV**

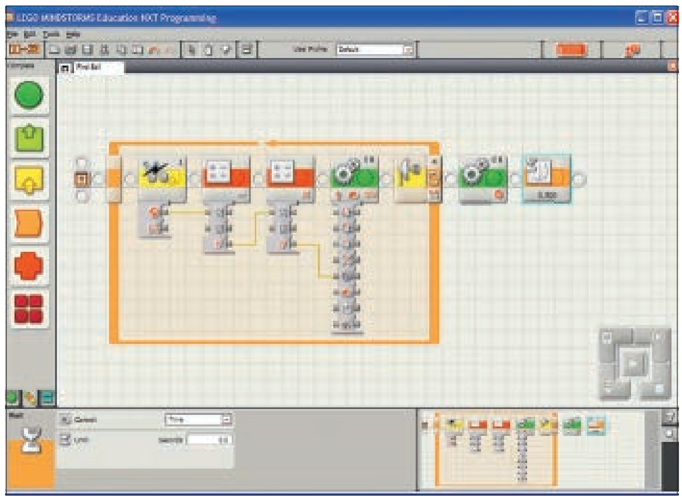


Figure 7. Add a Timer block and set it to .5 seconds.

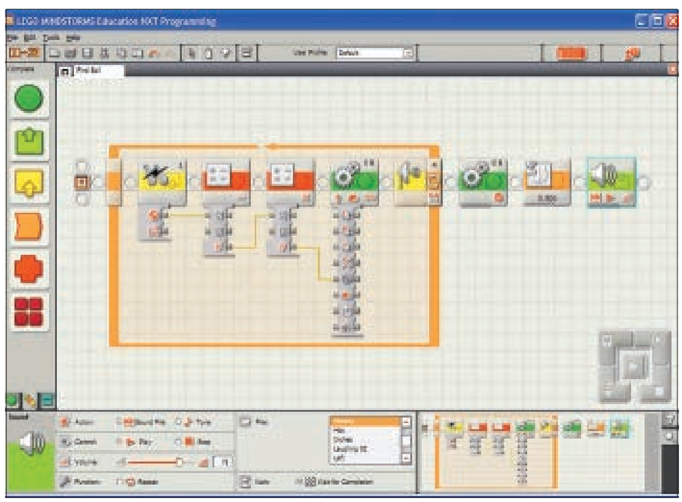


Figure 8. Finally, bring in a Sound block and select the sound of your choice!

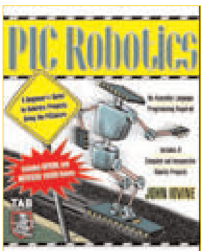
The *SERVO* Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS

ROBOTICS

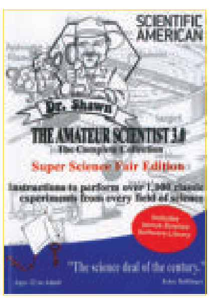
PIC Robotics
by John Iovine



Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU!

In this heavily-illustrated resource, author John Iovine provides plans and complete parts lists for 11 easy-to-build robots each with a PICMicro "brain." The expertly written coverage of the PIC Basic Computer makes programming a snap – and lots of fun.
$24.95

The Amateur Scientist 4.0 The Complete Collection
by Bright Science, LLC



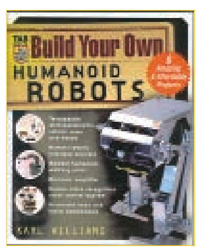
There are 1,000 projects on this CD, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!
Reg $26.95 Sale Price $23.95

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists
NEW!
by Dustyn Roberts

In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.
$29.95*

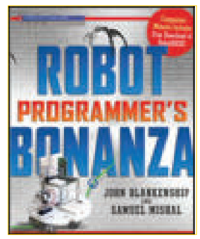
Build Your Own Humanoid Robots
by Karl Williams

GREAT 'DROIDS, INDEED!



This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.
$24.95*

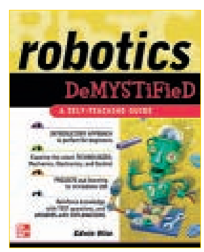
Robot Programmer's Bonanza
by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.
$29.95

Robotics Demystified
by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!

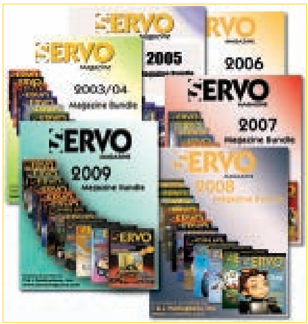


Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles



Published by T & L Publications, Inc.

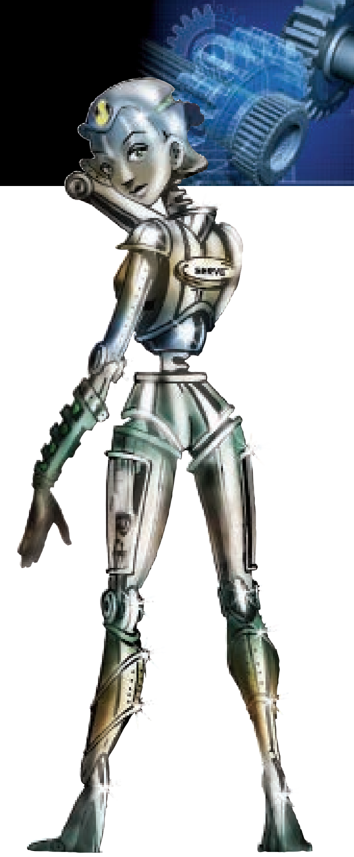
$57
per bundle

Save $10 off the normal price!!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

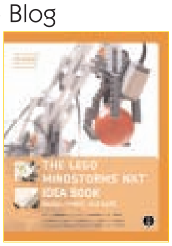
Call my Webstore and you'll get someone in AMERICA!

Visit my online store @ www.servomagazine.com



LEGO MINDSTORMS NXT Idea Book

by the Contributors to The NXT Step Blog

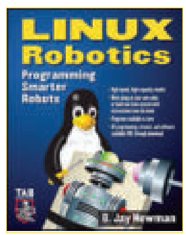


If you're serious about having fun with LEGO® robotics, you've come to the right place. The team behind The NXT STEP blog — the authoritative online source for MINDSTORMS® NXT information and advice — has packaged its considerable skills and experience in this book. Inside, you'll find some of the team's best ideas for creating cool and sophisticated models, including instructions for eight robots you can build yourself.

Reg $29.95 Sale Price $24.95

Linux Robotics

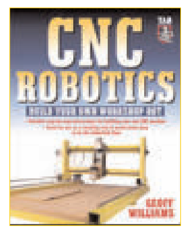
by D. Jay Newman



If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need. **$34.95**

CNC Robotics

by Geoff Williams

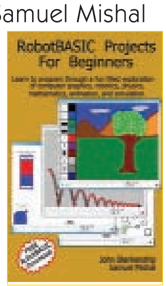


Here's the FIRST book to offer step-by-step guidelines that walk the reader through the entire process of building a CNC (Computer Numerical Control) machine from start to finish. Using inexpensive, off-the-shelf parts, readers can build CNC machines with true industrial shop applications such as machining, routing, and cutting — at a fraction of what it would cost to purchase one. Great for anyone who wants to automate a task in their home shop or small business. **$34.95**

SPECIAL OFFERS

RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal



If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Reg. Price $14.95 Sale Price $9.95

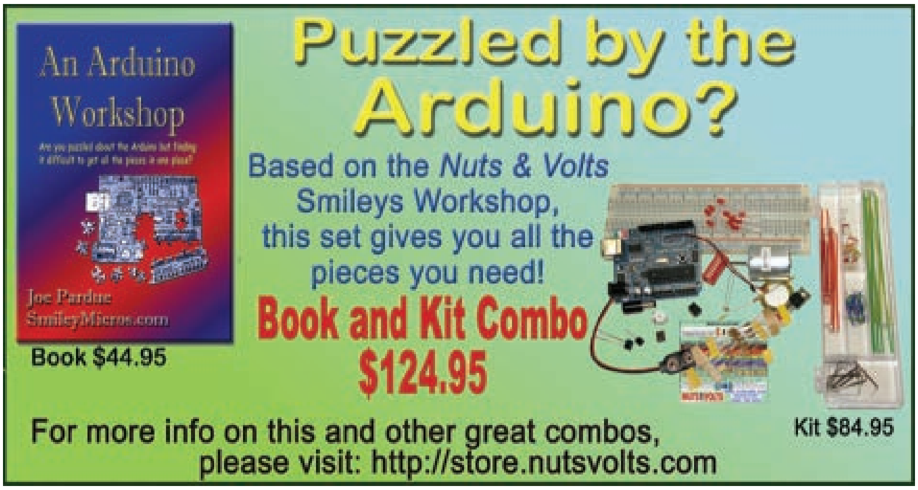
Technology Education Package for Everyone Starting in Electronics

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

Regular Price $79.95 **Subscriber's Price $75.95**

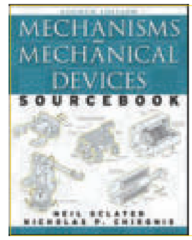
Or order online www.servomagazine.com

SPECIAL OFFERS



Mechanisms and Mechanical Devices Sourcebook

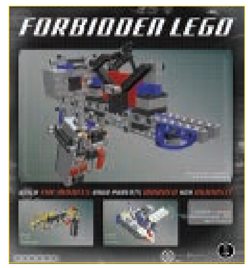
by Neil Sclater, Nicholas Chironis



Over 2,000 drawings make this sourcebook a gold mine of information for learning and innovating in mechanical design. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology will get you up to speed on these cutting-edge technologies. Easy-to-read tutorial chapters on the basics of mechanisms and motion control will introduce those subjects to you. **Reg $89.95 Sale Price $69.95**

Forbidden LEGO

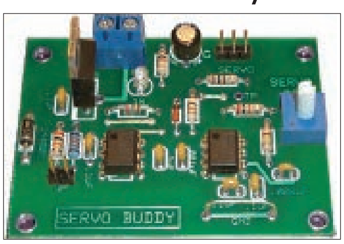
by Ulrik Pilegaard / Mike Dooley



Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse. **Reg $24.95 Sale Price $19.95**

PROJECTS

The SERVO Buddy Kit

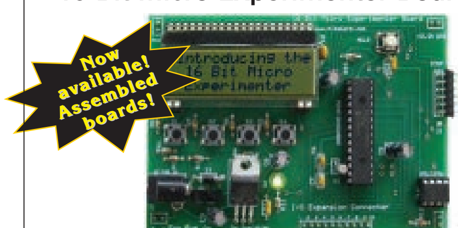


An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.
Subscriber's Price **$39.55**
Non-Subscriber's Price **$43.95**

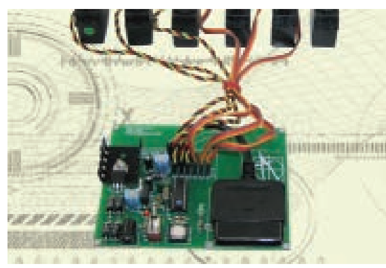
16-Bit Micro Experimenter Board



Ready to move on from eight-bit to 16-bit microcontrollers? Well, you're in luck! In the December 2009 *Nuts & Volts* issue, you're introduced to the 16-Bit Micro Experimenter. The kit comes with a CD-ROM that contains details on assembly, operation, as well as an assortment of ready-made applications. New applications will be added in upcoming months.

Subscriber's Price **$55.95**
Non-Subscriber's Price **$59.95**

PS2 Servomotor Controller Kit

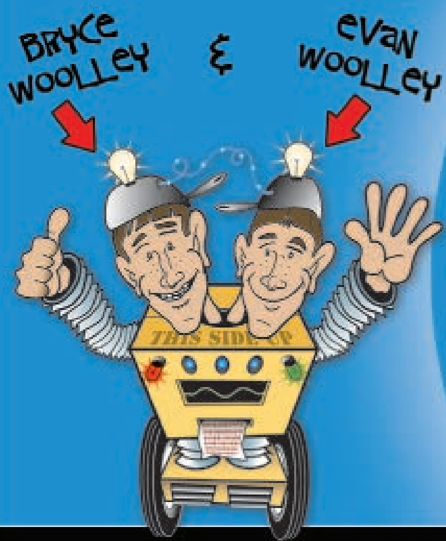


This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

For more information, please see the February 2011 edition of *SERVO Magazine*. Assembled units available!

Subscriber's Price
$79.95
Non-Subscriber's Price
$84.95

Twin Tweaks



THIS MONTH: *Mars Needs Motherships*

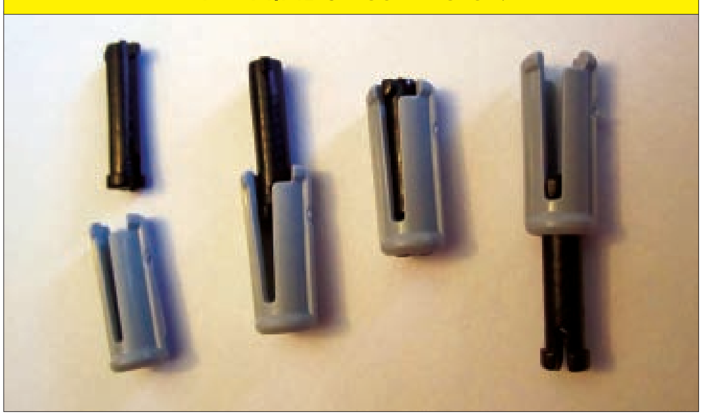
THE LUNAR ROVER 3-IN-1 PARTS.

This month, we have the pleasure of presenting the 3-in-1 Lunar Rover kit from MINDS-i. The MINDS-i system — invented by Mike Marzetta — is meant to be an expandable robotics platform that is more robust, powerful, and interchangeable than current offerings. The Lunar Rover itself is a large four wheeled radio-controlled vehicle with rack and pinion steering topped by a sleek solar panel. The 3-in-1 aspect of the kit is that the instruction manual suggests three different configurations for the base of the robot, ranging from a low profile bot suspended on leaf springs to a flexible bot equipped with colorful shocks. Having worked with a range of other robotics kits, we were eager to see what fresh ideas MINDS-i brings to the table.

The Light Side of the Moon

One of the first things you notice about the Lunar Rover kit is the scale. The approximately five inch diameter wheels are the first indication that you're dealing with something truly able to rove over some rough terrain, and the large kit of parts promises fertile ground for independent designs. The large solar panel also promises plenty of power for even large bots. The kit comes with an instruction manual and the website offers some helpful instructional videos that we recommend perusing before proceeding.

THE MINDS-I CONNECTOR.

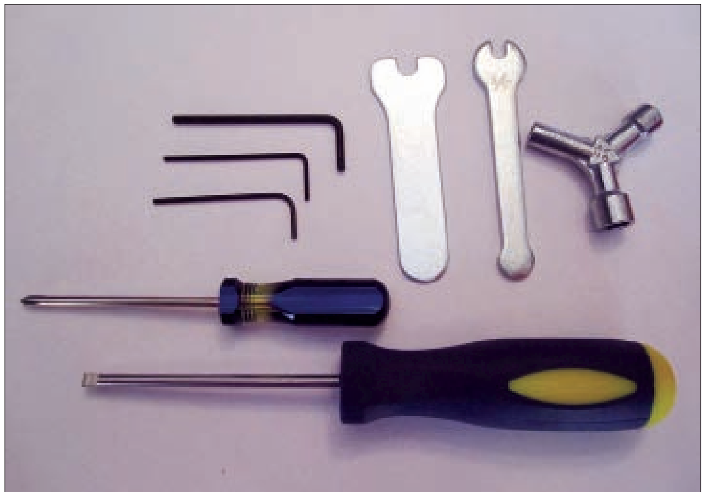


One of the numerous innovations of the MINDS-i system is the design of the connectors that hold the frame together. The connectors are comprised of an inner shaft and an outer shell, and operate in a way reminiscent of the cam bolts one uses to assemble do-it-yourself furniture. The core must be pressed into the shaft, and after insertion into the frame the core is turned 60 degrees to lock into the shell. The connectors come in four different flavors — two different lengths, and locking or smooth connectors. The locking connectors constrain the movement of the beams while the smooth connectors allow the connected beams to rotate. Unlike many kits, the Lunar Rover comes with its own set of tools which includes Allen wrenches and a special slotted screwdriver to use with the connectors. The screwdriver has a nib at the end of it that allows the cores of the connectors to be easily removed once pressed in all the way.

With three configurations to choose from, it was tough to decide which incarnation of the rover to build first. We settled on what seemed to be the second configuration. This appeared to be the one with the highest profile which we thought would make it easier to go back and add structural embellishments. The second configuration also

used the eye-catching shocks instead of the unassuming leaf springs for suspension, and we couldn't pass up the cool factor.

The manual recommends putting together the gearboxes and drive shafts first, but we started with the frame so we could get acquainted with the basic parts. Using the connectors and frame bits is very straightforward, and even though we had some initial worries about possible loosening and falling apart, once assembled the frame never gave us any problems.


TOOLS OF THE TRADE.

Embattled in Los Angeles

After crafting what looked like it could have been the start to a sturdy bridge, we turned to the locomotive elements of the bot. Even though the Lunar Rover is a four wheel drive robot, it uses only one DC motor and transmits power to the wheels through drive shafts and differentials. The gearbox for the motor is a very simple assembly, and even though the kit comes with a motor, the folks at MINDS-i tout the rover as compatible with virtually any hobby motor. Such a bold statement is backed up by the adjustability designed into the gearbox – a slotted mount allows the builder to achieve the ideal mesh between the gears with the provided motor, but also would accommodate motors of numerous other sizes. The included motor is a 5,000 RPM DC motor that promises plenty of power thanks to a 5.8:1 gear ratio.

The kit offers two options for the differentials: a traditional differential and a fixed differential. The instruction manual does not recommend one differential over the other, though we thought that the traditional differential would be the better choice for an intrepid rover. The fixed differential would cause inefficient slipping as the rover turns and climbs over terrain. The fixed differential would be better for speed on a straightaway, so we could go back and add it later if we were in the mood to make a lunar drag racer. The spider and pinion gears for the differential are impeccably machined, and fit perfectly in the pumpkin.

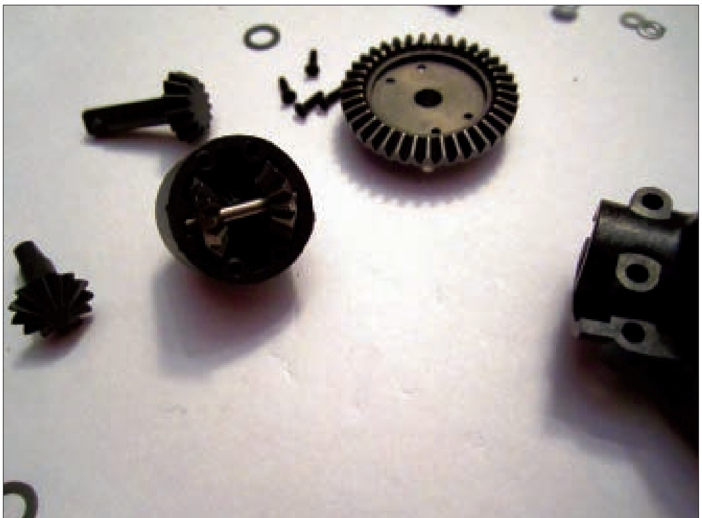
The kit wisely recommends for the user to keep track of the side that the ring gear is on before hiding it in the outer casing, and we took the suggestion of identifying the gear side by the heads of the screws holding the casing together. The differential spins nicely, as long as you don't overtighten the screws holding the casing together. The casing is held together only by two screws near the opening for the shaft of the pinion gear. Tightening these screws pinches the casing around the bearings that house the shaft, and opens the casing on the opposite end from the screws. This devilish pinching seems to cause the ring gear to rub against the inside of the casing, making the differential run very roughly.

Luckily, this problem has an easy solution. When installing the differential in the robot itself, it sits in a frame that surrounds the casing nicely. Thus, even leaving the screws on the casing loose runs no risk of the differential falling apart. The frame keeps everything together well and the differential running smoothly.

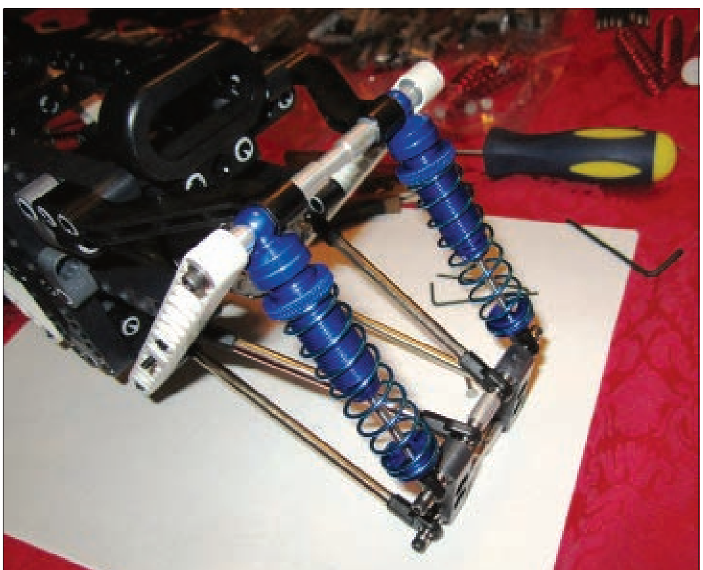
One of the themes running throughout the kit is the necessity of cutting threads into a lot of the plastic parts as you screw them together. This is an aspect of the kit that we had a bit of a love/hate relationship with. On the one hand, we appreciate the fact that this adds to the durability of the kit. Threading the plastic yourself acts sort of like a Nylock nut and holds the parts together better. But threading the screw in the first place can be a test of endurance for your fingertips, particularly when all you have to grip onto is the thin metal handle of an Allen wrench. The task also demands steely resolve, because restarting runs the risk of cross-threading the part. Being cognizant of the parts that need to be threaded is really all that's necessary, as long as you are prepared to apply the necessary pressure to begin with. Some parts like the U

THE HEART OF THE ROVER.

Step-by-step progress.



Assembling the differential.



Cool shocks!

brackets for the drive shafts are even being redesigned to make the task of threading the screws easier.

Speaking of the U brackets (or yokes, to use the term of the instruction manual), that brings us to the most devilish part of the kit. The instruction manual and video on the website indicate that installing the ball joints into the U brackets should be an easy procedure; one that simply requires a minor application of force using the shaft of a screwdriver. So simple that it could be done with one hand. This, however, was not the case. Absent any modification, superhuman force was required to fit the ball joints into the U bracket. We're pretty sure the screwdriver would have broken before the stubborn plastic yielded to the ball joint. We were surprised by the resistance, especially given the design of the U brackets. A shallow channel is visible that appears to gently lead the nibs on the ball joint into its proper resting place. The shallow channel, however, is not enough to compensate for the thick, rigid plastic that makes up the U bracket.

Forced to get creative, we tracked down a small taper file to deepen the channel. Despite some initial doubts of the efficacy of the hack, after a bit of filing the ball joints did eventually submit to pressure from the screwdriver. It was still not an easy task, but it was most definitely physically possible.

Sunshine on the Skyline

With the drive shafts assembled, we were confident that the rest of the construction would be comparatively easy. Our hopes were realized as the wheel assemblies attached to the rest of the frame, and the kit began looking like a real rover. After test-fitting the solar panel, all we had to do was install the electronics.

The instruction manual provides a helpful diagram for wiring up the robot which is a refreshingly straightforward affair. The electronic components include a small speed controller, the battery pack, the receiver, the on/off switch, the wires to the motor, and the solar panel converter. The wires for the motor are outfitted with sockets, so no soldering is required. What is required, however, is a bit of creativity while placing the electronic nervous system into the robot.

The instruction manual tells you how to build a killer drive base, but it provides no suggestions or attachment points for the electronics. Perhaps that is a bit ungenerous – the kit includes some double-sided tape and zip ties, and instructs the user to have at it, and the frame does include a handy casing for the battery. The double-sided tape is actually super helpful, and even though the amount seems small it is more than enough to stick the speed controller, receiver, and on/off switch to the frame.

The rover was finally completed. Given the difficulties with the initial assembly, one might think that the 3-in-1 nature of the kit might be a cause for concern rather than celebration. The most difficult aspects of assembly, however, are the drive shafts, and those only need to be

put together once. Other than that, the kit can be easily disassembled with no more than the MINDS-i tool, a Phillips screwdriver (one is supplied with the kit), and a few Allen wrenches. The MINDS-i tool in particular makes disassembly of the frame rather rapid, and the quick action of pulling out the connector cores is enough to make you feel like a pneumatic tool wielding member of a speedy pit crew.

The kit also comes with a charger for the battery so that users won't be left in the lurch on a cloudy day. With a fully charged battery, we were ready to test the rover and see if the big wheels and shocks could handle all of the terrain that we imagined it could. We were not disappointed. No matter what we placed in its way — whether it was feet or casebooks — the rover always effortlessly bounced over every obstacle.

THE HARD EARNED DIFFERENTIAL.

THE OBSTINATE U BRACKETS.

Dependence Day

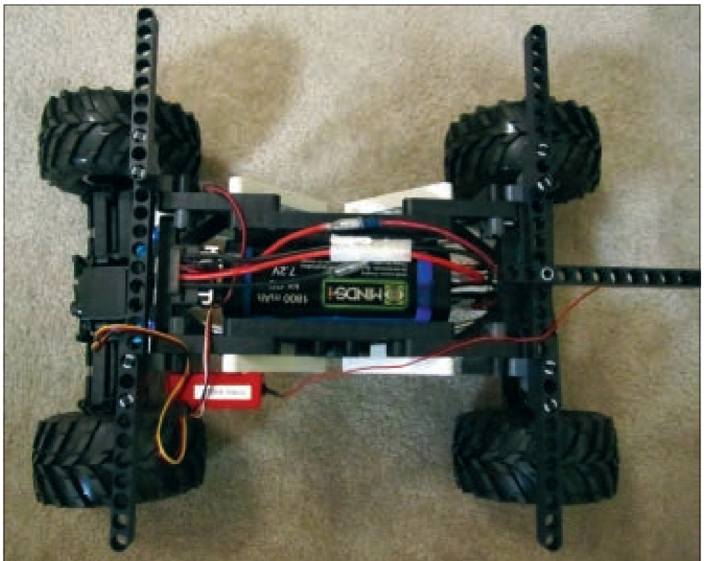
The solar charge controller converts the voltage and current from the 13 watt solar panel into something suitable for charging the 7.2V rechargeable battery pack. The controller is easy to wire up because all you have to do is connect one end to the solar panel and one end to the battery back. To prevent any disastrous wiring mix-up, one end uses pins and the other uses a socket. The much thicker gauge wire on one end is also an easy way to tell which side plugs into the solar panel.

We were delighted by the ease of implementing the super cool solar panel but we noticed that because there was only one hookup to the battery and no type of splitter, the battery could only be charged when it was not actually hooked up to the rest of the robot. A way to take advantage of the size of the Lunar Rover while making use of the solar panel would be to give the rover something to charge while roving around. Even though the rover was an appropriate size for a nurturing mothership, we still needed something fairly compact to be carried around. The challenge was to find something small that could be charged by the hefty solar panel – ideally we wanted to find something with a 7.2V rechargeable battery pack just like the rover itself. It just so happened that our trusty VEX kit also used a 7.2V rechargeable battery pack, and we were capable of making a pint-sized robot out of the VEX parts.

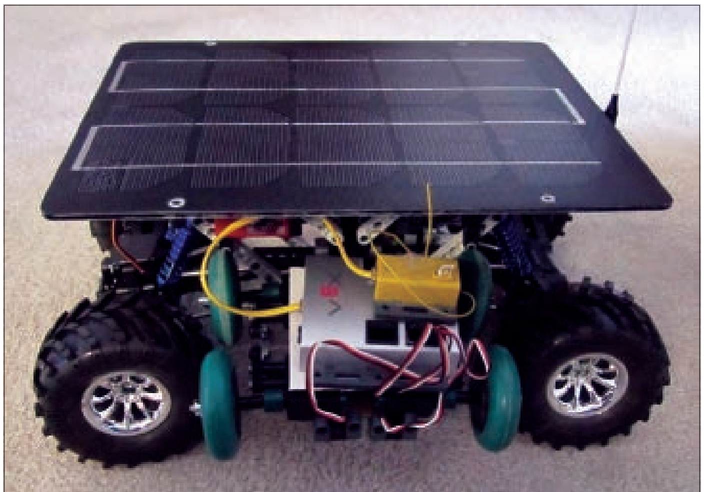
We also thought that this was a nice opportunity for the inevitable comparisons drawn between different robotics kits. Putting together the small VEX robot was a much quicker exercise than the assembly of the rover. This was likely in part due to familiarity, but also by the inherent ease of using classic screws and lock nuts. That ease of assembly had a price of course, because throughout our testing of the VEX robot we had to periodically tighten the screws. Despite testing the rover over a variety of obstacles, the MINDS-i bot needed no additional tightening.

Perhaps the distinction was exaggerated by our goal of creating a tiny robot with the VEX kit, but the scale of the kits is one of the most notable differences. Correlated to the scale of the kits is the power used to move them. The VEX robot is great for small scale tasks — like the object retrieval of Science Olympiads of yore — but even a major gear ratio will give the VEX servos only limited oomph. Also, the specialized VEX parts make implementation of non-kit motors an option, but one that requires extra hacking. The Lunar Rover, on the other hand, is specially

THE COMPLETED DRIVING BASE.



WIRING UP THE ROVER.



BEING A GOOD MOTHERSHIP.

CHARGING THE VEX ROBOT.

designed to be compatible with non-kit motors.

We had all the parts we needed to turn our Lunar Rover into a Lunar Mothership. A simple shelf was all that was needed to hold our VEX robot which was small enough to fit perfectly between the wheels of the rover. Despite its small size, the VEX robot still had some weight to it, so we needed a counterbalance on the other side of the rover. After numerous attempts including coasters, water bottles, and a clock, we found the ballast with the ideal size and weight: a glass of Starbucks vanilla frappuccino. With our tiny VEX robot nestled away, we had a robot within a robot. It was a good thing that we didn't have to go deeper, because a robot within a robot within a robot would have been a bit much.

Even carrying around a VEX robot and a frappuccino, the rover was able to negotiate obstacles with ease and didn't seem to lose any speed. Such success indicated the rover would be more than able to carry around a variety of mechanisms, made both from kit and non-kit parts. One of the limitations on expandability that many kits face is the lack of an intuitive way to include non-kit parts in a project. Most kits use their own kind of fasteners, and the MINDS-i system is no exception with its two-part connectors. And while these plastic connectors might be a great and easy way to put the kit parts together, most non-kit parts rely on far more traditional nuts and screws which can be difficult to mate to the specialized plastic frame bits.

The MINDS-i kit offers an elegant solution to this problem by including screw adapters that fit into the frame mounting holes. The screw adapters come in two halves, with one half being a through hole and the other featuring a narrower bore that 4-40 screws can thread into. The screw adapters are used in the kit as a way to attach the shocks, and we also see them as an easy way to add non-kit parts to the rover.

Opening Up Your Mind's Eye

We think the folks at MINDS-i are on track with their goal of creating a system more durable, powerful, and interchangeable than many current offerings. On the note of durability, many kits that are easy to put together are also easy to take apart – by the user and by things like terrain and simple usage. While the Lunar Rover took some effort and force to assemble, the result was something that wouldn't fall apart when you didn't want it to. Also, the most difficult parts to put together were the ones that you wouldn't have to disassemble again, like the drive shafts.

On the note of power, the single DC motor in the Lunar Rover gives the robot impressive speed and more than enough torque to clamber over all sorts of obstacles. Perhaps the most exciting aspect of the kit, however, is the interchangeability. The MINDS-i system is designed to work with virtually any hobby motor, and thus allows users to power their rovers (and any additional mechanisms) with even beefier motors. This was quite appealing to us in particular, because over the years we have found other

robotic design systems to be well suited to a variety of robotics competitions with one major caveat – the kit motors for many of these kits are not powerful enough to handle arduous tasks. Also, the receptiveness of the MINDS-i Team to customer feedback and the actual implementation of that feedback in design changes is the hallmark of a great robotics system. The devilish U brackets and a few other plastic parts have been redesigned in response to customer feedback. The two major issues identified above – the difficulty of threading some of the parts yourself and the challenge of installing the ball joints in the U brackets – have been addressed by new molds for some parts. With such a responsive team behind the MINDS-i system, we can envision kits like the Lunar Rover becoming favorites in competitions and in the classroom.

Some purist roboticists may still not be impressed. They demand to see a platform that is more than a nice construction system, but one that can be programmed and outfitted with sensors. The MINDS-i folks have planned for just such an audience, because they've partnered with PCS Edventures to create a comprehensive robotics design system. The Lunar Rover kit and others are designed to be compatible with the PCS Brain microcontroller and sensor pack. Far more than just a cool way to add some autonomy to the MINDS-i kit, PCS Edventures has designed curriculum meant to help teach robotics in the classroom. We'll get the chance ourselves to test out the PCS Brain in the Lunar Rover in an upcoming article, and we hope that you join us for that "edventure!" **SV**

RECOMMENDED WEBSITES

www.mymindsi.com
pcsedu.com

LUNAR ROVING!

Then and NOW

THE NEXT STEP IN ROBOTICS

by Tom Carroll

Back in February '11, I followed a series of threads on the Seattle Robotics Society's list server that concerned the next step in robotics. Many people from all over added their comments and — as with most conversations within these various groups — the subject began to stray a bit after a week or so. It was quite interesting to see the many viewpoints expressed. There are some absolutely brilliant robot experimenters in this world. Needless to say, the basic concepts mentioned concerned robots of the experimental or hobbyist variety.

To start the conversations, someone had asked "What is the next step <in robotics>?" As with many questions that I have discussed over the years, a question such as this is highly dependent upon the person who is asking the question. Is the person a robotics hobbyist or experimenter, a university student working on a thesis project or a micromouse (researcher designing a robot to assist the elderly)? Is it a person who is interested in only humanoid bipedal robots, UAVs, space-borne robots, military robots, or any of the many other types? The next step could have widely different meanings to different people. For the sake of this discussion, I am going to concentrate on just a few types of experimenter's robots that might be used in contests or built just for fun and education.

Though everyone seemed to have their own idea of exactly what the 'next step' may be, quite a bit of the feedback seemed to center on software approaches, though some looked to the future of the overall market for robots. One extremely optimistic outlook written in 2009 for the market of personal robots — in particular, elder care robots — saw the market in 2010 to be $74 billion and growing to $86.6 in 2014. A Robotics Trends' article by ABI Research written just last December saw the market for the complete realm of personal robots to be a bit more modest at $19 billion in 2017. This figure included telepresence and security robots, as well as health care, business, commercial, and the many varieties of home robots. As reality has set in with this recession, it is hard to believe even these last figures, but we all remain hopeful. Most robot companies would not mind having a few billion dollars tossed into the ring right now.

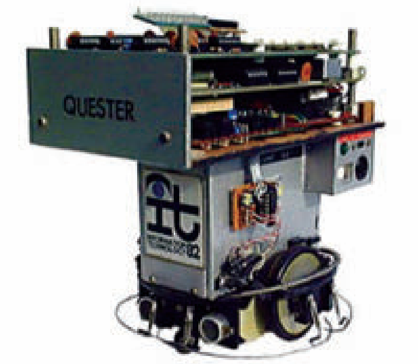


FIGURE 1. David Buckley's 1981 Quester Micromouse.

The Next Step in a Micromouse

I've selected the IEEE Micromouse contests and several variations as an example of a widely recognized and long-running series of contests to illustrate how a series of next steps have resulted in dramatically newer 'mice.' Back three decades ago in 1981 when the IEEE Micromouse contests had been around for only a few years, David Buckley of the UK built his unique Quester shown in **Figure 1** for the contest. The various Micromouse contests use a maze up to 20 feet square consisting of 6-1/2 inch wide pathways with two inch walls as shown in **Figure 2**. I remember seeing articles about Buckley's rather large mouse from the UK. David has built more robots than almost anybody that I know of and Quester certainly turned some heads back in the '80s. I'll use his machine as a sort of starting point for many next-step robots. You can read much more about Quester and many other robots at David's website at **davidbuckley.net**.

Quester's Specifications

Let's take a look at some of

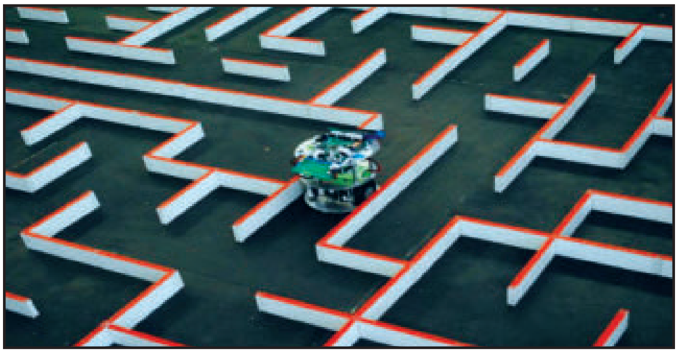


FIGURE 2. A typical Micromouse maze in 2010.

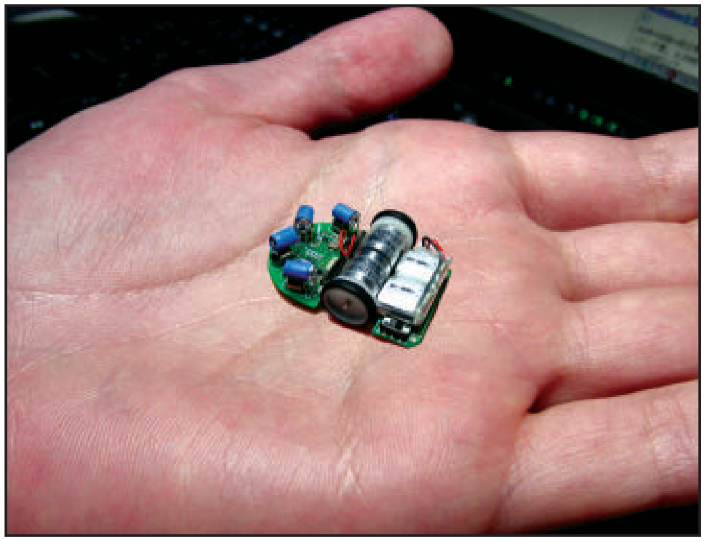


FIGURE 3. A quarter scale Bee Micromouse robot from Japan.

Quester's features:

- Differential steering with quadrature shaft encoders on each wheel. Buckley cut a set of evenly spaced holes in the output gears and used sets of mini LEDs and phototransistor receptors for the wheel encoders. (Note: Many early Micromice started out with Ackermann or 'car-type' steering but found turning corners very difficult).
- Distant obstacle detection using a pair of forward-facing ultrasonic Tx & Rx transducers.
- Visual sensors are two compounds eyes composed of three phototransistors each, with a narrow field of view for each individual phototransistor to detect walls, side openings, and pathways.
- The processor is the 6502 — a popular microprocessor used in many computers of that time, including the early single-board computers KIM-1, Sym-1, and the Rockwell AIM 65. Later, it found use in the very popular Apple II. The processor board has a 512 byte monitor, an LED readout, hex keypad, and a mighty 1K of RAM. Programming was via a tape recorder interface. David had to manually assemble the code to make the 1K barely sufficient to map and run the maze. David states that his processor board back then cost the equivalent of one of today's high-end computers. He later fitted the robot with an arm and gripper, though the initial processor was inadequate to handle any more peripherals. The robot is still operable to this day, although the tape drive and its programs no longer work.

What would David have imagined that the next step might have been when he was back in 1981? As he stated in February's comments on the SRS list server: "Today's control computers are much more powerful and a heck of a lot easier to use, but the software ideas for small robots haven't moved on. My robots now have a multilevel behavioral operating system (BOS) with reactive subsystems for survival and human interaction, sequence subsystems so they can be given lists of things to do, and deliberating subsystems so they can sort out internal conflicts. Above all, they accept high level commands so I don't have to keep editing the program."

Micromice of Today

Micromice — as well as almost all personal, experimental, and hobby robots — have changed quite a bit in the past three decades. The 1/4 scale Bee shown in **Figure 3** from **robots-dreams.com** is probably the next few steps beyond Quester and is not at all typical of most micromice in today's contests. The dramatic reduction in size over the years illustrates the many next steps that competitors have used in construction of their robots.

The Bee's Specs and Components

Check out these top features:

- The Bee's top speed: Two meters a second or 6'7"/sec — really fast.
- Epson Toyocom XV-3500CB gyro (a great gyro).
- Two Nemicon ultra micro encoders OMS-125-2T.
- Two full-bridge motor drivers ±2.8A 36V.
- A3950 STMicroelectronics ARM Cortex-M3 processor ($4.50).
- 70 mAh Zippy lithium polymer batteries.

Japan has been very supportive of Micromouse contests for over 27 years. The Bee and another half-size mouse shown in **Figure 4** made by RT Corporation in Japan are typical of these smaller competitors. As you can see, the Bee is a very tiny robot and is a quite capable competitor with speeds of a real mouse scurrying away from a hungry cat. The larger — but still tiny half-size mouse next to a nine volt battery — appears to be about a two inch cube. The larger AIRAT-2 shown in **Figure 5** from the Hong Kong-based Robot Store is another more typical Micromouse robot. It uses an 8051 (remember those?) CPU on a JS8051-A2 processor board, and appears well crafted with nicely machined wheels and an LCD readout on the CPU board. It's well laid out and costs $650.

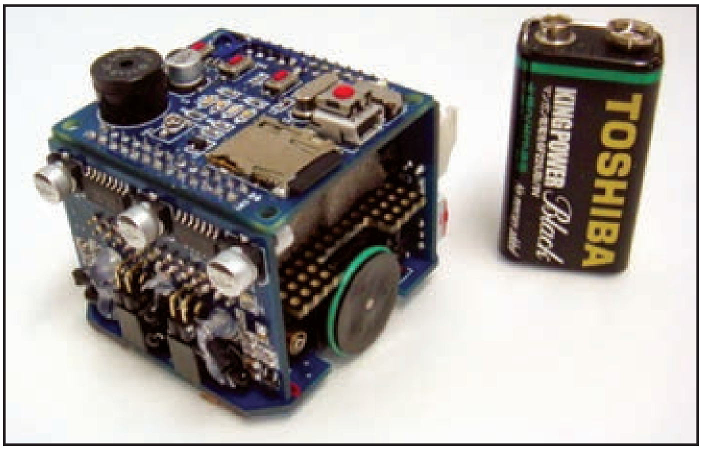


FIGURE 4. RT Corporation's half-size Micromouse kit.

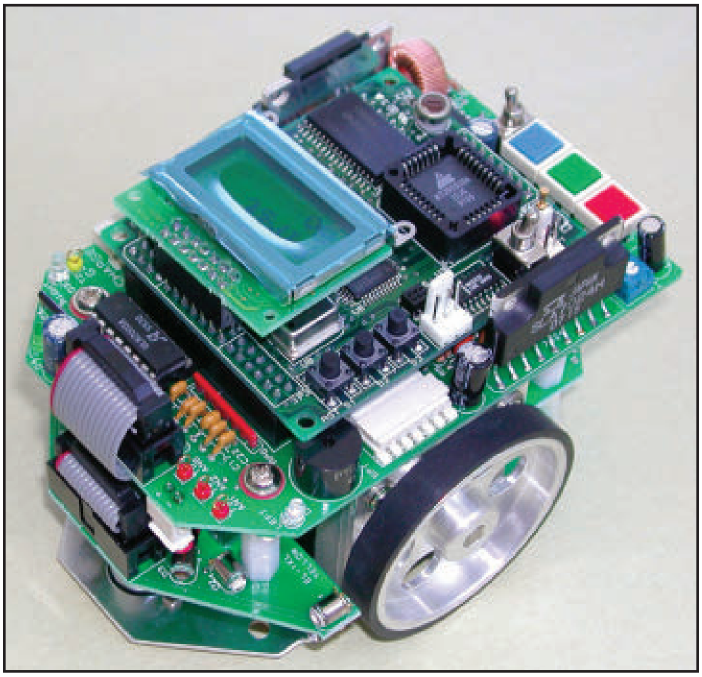


FIGURE 5. Robot Store Hong Kong Micromouse for $695.

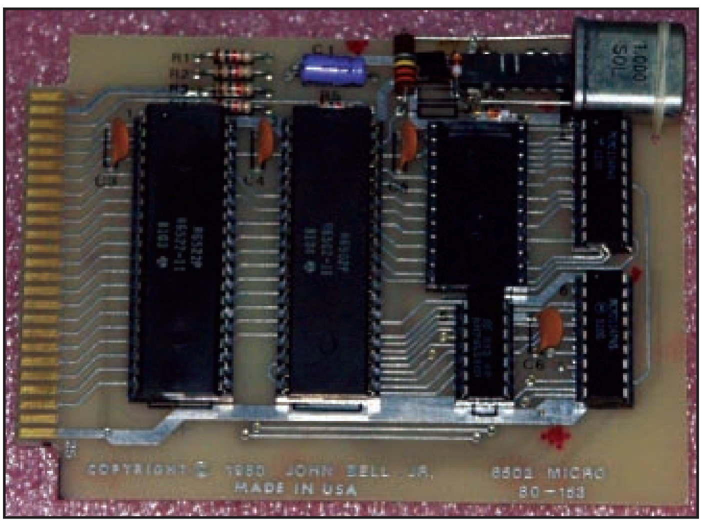


FIGURE 6. The John Bell 6502 microcomputer board.

The Next Steps in a Microcontroller/Processor

We've seen how the physical structure of Micromouse robots shrunk over the years, as well as how processing and sensor power has increased. It is the various processors that we've used over the years that allow our robots to take the next step in capabilities. Back in the '80s, a popular single-board computer used in many experimental robots was the John Bell Engineering 6502 Microcomputer. The Model 80-153 board consisted of a 40-pin DIP 6502 seen as the center large IC in **Figure 6** (by Don Sawyer), as well as a 6522 peripheral interface adapter. This is the same processor used in Buckley's Quester, as well as Apple and many other personal computers. The board had 1K of RAM and 1K of ROM, and the company also sold an EPROM programmer that was quite handy for loading and changing programs. I used two of them in one of my early robots.

In the mid '80s, Motorola (now known as Freescale) developed the 68HC11 eight-bit microcontroller that is still used in many products from bar code readers to robotics applications. The microcontroller was the logical next step to simplify control system designs as these devices only had to communicate between sensors and controllers — not humans. Descended from the 6800, this next step chip is available in a 48-pin DIP and SMT versions, and there are many varieties that include the "Buffalo" bootloader and an EPROM. Popular through the '90s, the 68HC11 has lost favor with robot builders to the newer microcontrollers and smaller microcontroller boards.

The BASIC Stamp Leads the Way

The BASIC Stamp by Parallax (shown in **Figure 7**) is probably the largest next step made in experimental and hobby robotics. Brought out in the '90s, this tiny board is based on the Microchip PIC16F57 microcontroller. It is not necessarily the best overall solution for robotics experimenters but it is undoubtedly the most popular by far, though the Arduino series of microcontroller boards are trying to catch up. The BASIC Stamp single-board computer became instantly popular among engineers and hobbyists due to its ease of use. It runs Parallax's PBASIC language interpreter and the developer's code is stored in an EEPROM. The 24-pin DIP package (for the BS II) has branched into many versions over the years, and Parallax makes what they call the Board of Education — a developer's board on which a user has a breadboard area for project development. This board is the basis for many classroom educational projects and their BOE-Bot robot.

Some potential users have hesitated in the use of the Stamp since its price starts at $49 (for the BS II) and upwards for other versions. Many less expensive clones have been made by other companies, but Parallax has great customer service, many versions and accessories available

for its products, and a large amount of documentation available on the Stamps. One of Parallax's latest chips is the multi-processing Propeller — a chip with 32-bit architecture containing eight processors (cogs) operating at up to 20 MIPS, 32K of RAM and ROM, and utilizing Spin or assembly languages. At less than $8, it has been incorporated into many robot projects. There is a Propeller demo board for $60 and a proto board including the chip available for $25. That's a lot of processing power available for less than the Stamp II.

Parallax also carries a full line of sensors from compasses to sonars to GPS modules, and robots ranging from the popular $160 BOE-Bot to the $3,500 Quad Rover (shown in **Figure 8**) that is powered by a Honda gas engine and is Propeller based. Three and a half kilobucks for a robot might sound like a lot of money, but the 89 pound machine offers a lot for the serious experimenter. The air-cooled, four cycle 2.5 HP engine drives a hydraulic system for maneuvering the wheels. GPS, accelerometers, vision, and assorted navigation systems can be added for various experimenter's designs. Go to **parallax.com** for more information.

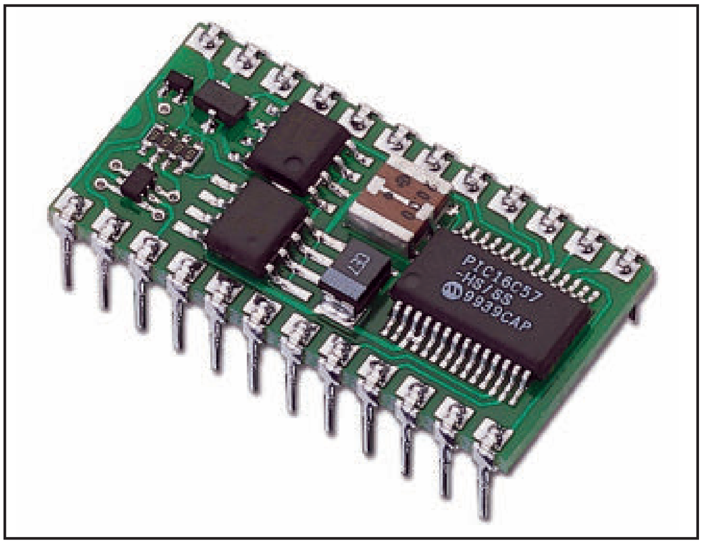


FIGURE 7 The BASIC Stamp II.

FIGURE 8 Parallax gas-powered Quad Rover.

Is the Arduino the Next Step After the BASIC Stamp?

The Arduino Duemilanove microcontroller board shown in **Figure 9** has become very popular since its introduction in 2009, due in part to its cost of about $25 and its open-source computing platform. For those who are curious about the name, "Duemilanove" means 2009 in Italian. Powered by the ATmega 168 running at 16 MHz, the board can operate off of 5 VDC from a USB connection or from external power. It has 16K of Flash memory, 1K of SRAM, 512 bytes of EEPROM, and 16 I/O pins. The newer Arduino Uno shown in **Figure 10** costs $30, uses the ATmega 328, and has 32K of Flash memory. Both boards use the Arduino programming language which can be downloaded free from their site. These boards are available at the Robot Shop, Pololu, and many other distributors.

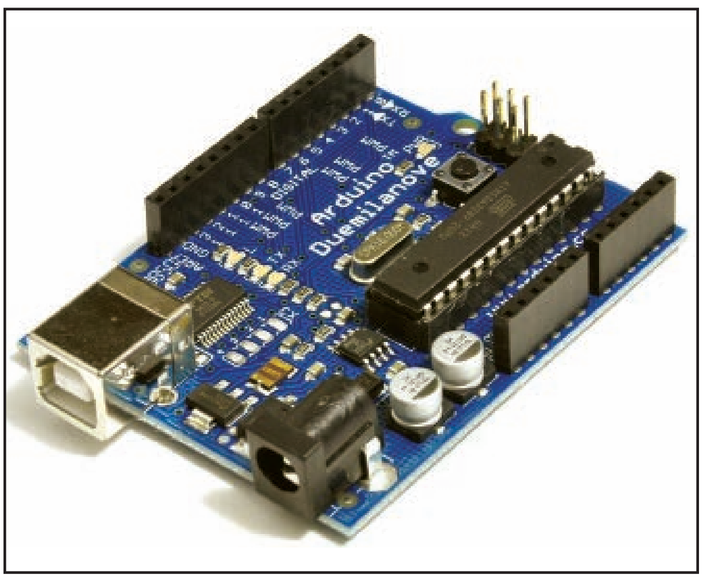


FIGURE 9. The Arduino Duemilanove microcontroller board.

New Technology Leads to New Robot Designs

Some might look to improvement in software as the next step in robots. The use of PID (Proportional Integral Derivative) loops in robot control is favored by an increasing number of experimenters but it is not new as it has been used in industrial control systems for years. LabView by National Instruments is another software approach that many robot experimenters have taken, especially using the NXT controller from LEGO and the Mindstorms kits.

The kit approach is an excellent next step for persons desiring to learn a particular robot technology or just getting a feel for robotics without cutting metal. The LEGO kits with the NXT microcontroller and VEX kits offer a wide variety of robot designs; many are applicable to advanced

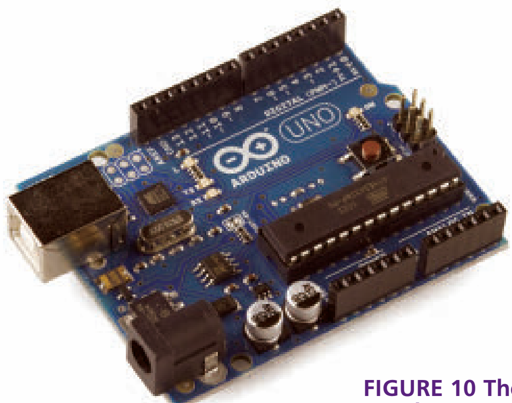


FIGURE 10 The newer Arduino Uno board.

studies. The newest VEX microcontroller shown in **Figure 11** (and priced at $250) uses the powerful STMicroelectronics ARM Cortex-M3 processor which — by coincidence — is the same processor used in the Bee Micromouse robot mentioned earlier.

Features of VEX's STM ARM-based Microcontroller

This microcontroller is wireless with built-in VEXnet technology and has:

- (8) Standard three-wire motor ports.
- (2) two-wire motor ports.
- A "smart sensor" port.
- UART serial ports.
- (8) Hi-res (12-bit) analog inputs.
- (12) Fast digital I/O ports which can be used as interrupts.
- Programmable with easyC v4 for Cortex or ROBOTC for Cortex and PIC.

I've just touched on a few of the key next steps by looking at the Micromouse robots and microcontrollers. Of course, there are a multitude of areas in this science that

FIGURE 11. The new VEX Cortex microcontroller.

one can extrapolate into more next steps. Basic robot mobility has progressed from Ackermann steering to differential steering to two-wheel gyro/accelerometer balancing to autonomous aerial and underwater vehicles, and even more. Battery power types have progressed from dry cells to lead acid to NiCad to NiMH to Li-Ion to LiPo (though all of these are still used), with different chemistries and technologies waiting in the sidelines.

Robot navigation has progressed from simple bumper switches to active light LED/phototransistor rangers to piezoelectric/electrostatic ultrasonic transducers to Lidar to GPS, and more. Even motors have progressed from basic wound DC motors to PM DC motors to rare earth PM to stepper motors to brushless to pancake to coreless and other types. People have said that the next step should be robots that can listen and understand speech, and that has been accomplished to a small degree as we'll discuss next. Experimenters around the world saw Honda's Asimo and wanted robots that would walk as bipeds; hundreds of varieties of those have now been built.

What is the Next Step?

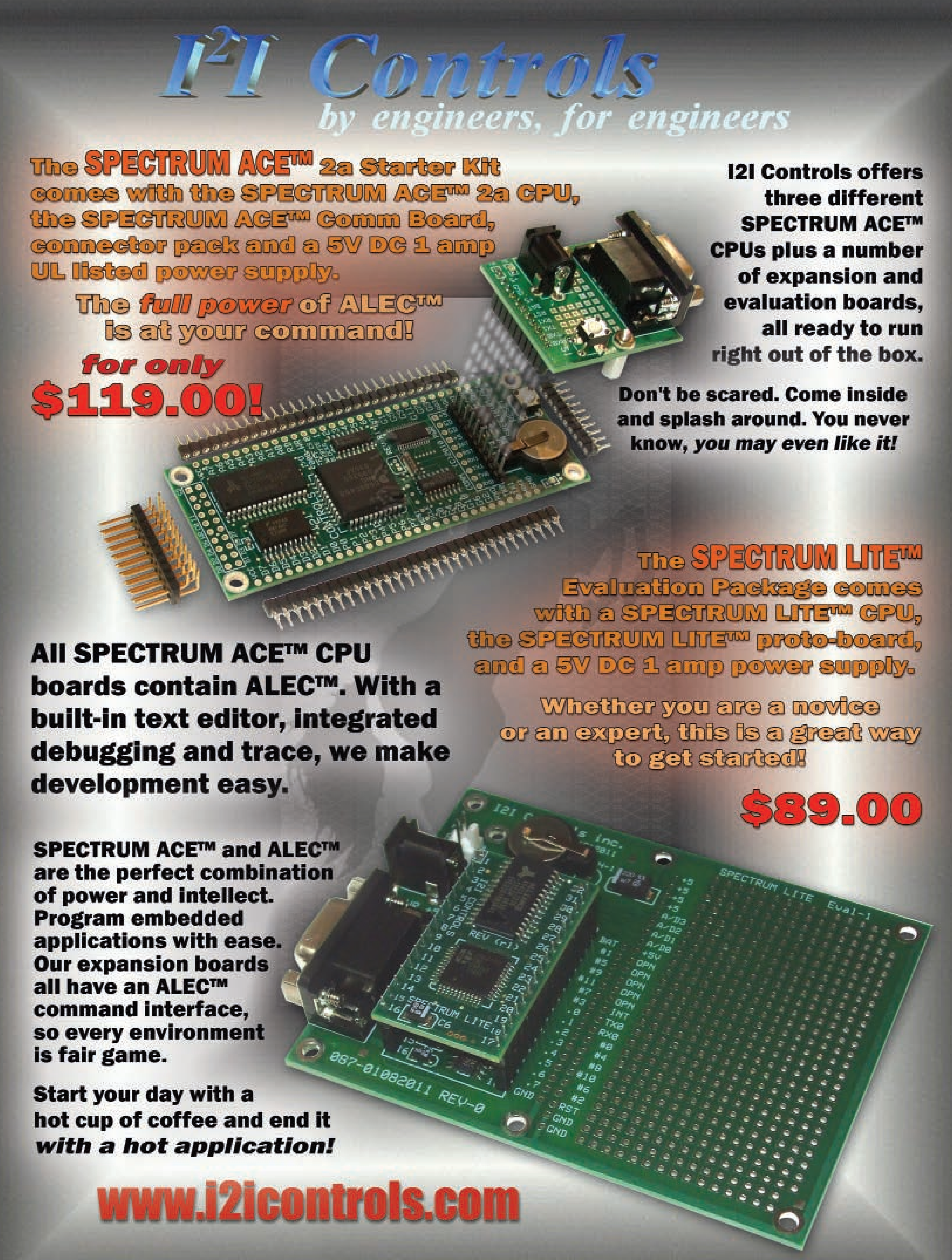
In Isaac Asimov's *I, Robot* series of short stories, Robbie was about a baby-sitter robot who could do everything that a human being could do but speak. Robbie could follow any commands and questions that his charge, little Gloria Weston, asked of him. I have always found it interesting that later in the story in a museum exhibit, people were amazed to see a robot who could speak; obviously the next step in robotics in this fictional story. These days, we can easily make our robot speak with many varieties of speech synthesizers and chip sets. However, speech understanding by a robot is still quite difficult for the average amateur builder — just the opposite from Asimov's story. Noise cancelling and directional microphones help a bit, but we're a long way from a truly understanding robot — though IBM'a Watson did a good job at Jeopardy. I highlight this example to emphasize that one person's next step is probably quite different from another's.

Final Thoughts

You might now be tempted to follow what different people are saying is the newest way to build a robot. Some very amazing revelations about the science are available to read, but you should also follow your own instincts after studying what others have said. You might have some heavy lead-acid batteries that take a full charge and are appropriate for your design. You may have found an old Z-80 processor board that still works fine with Basic. Perhaps you've got some power MOSFETs that will make a great H-bridge to drive your motors. Your old Polaroid electrostatic rangefinders, some old toy tank treads, and an Armatron robot arm are just waiting to be part of your new robot. So, go for it. Learn from it. You don't need the latest technology to build a great robot. SV

ROBO-LINKS

Advertiser Index


I2I Controls
by engineers, for engineers
The SPECTRUM ACE™ 2a Starter Kit comes with the SPECTRUM ACE™ 2a CPU, the SPECTRUM ACE™ Comm Board, connector pack and a 5V DC 1 amp UL listed power supply.
The full power of ALEC™ is at your command!
for only $119.00!
I2I Controls offers three different SPECTRUM ACE™ CPUs plus a number of expansion and evaluation boards, all ready to run right out of the box.
Don't be scared. Come inside and splash around. You never know, you may even like it!
The SPECTRUM LITE™ Evaluation Package comes with a SPECTRUM LITE™ CPU, the SPECTRUM LITE™ proto-board, and a 5V DC 1 amp power supply.
All SPECTRUM ACE™ CPU boards contain ALEC™. With a built-in text editor, integrated debugging and trace, we make development easy.
Whether you are a novice or an expert, this is a great way to get started!
$89.00
SPECTRUM ACE™ and ALEC™ are the perfect combination of power and intellect. Program embedded applications with ease. Our expansion boards all have an ALEC™ command interface, so every environment is fair game.
Start your day with a hot cup of coffee and end it with a hot application!
www.i2icontrols.com